

# Learning Unification-Based Natural Language Grammars

Miles Osborne

Submitted for the degree of Doctor of Philosophy

The Intelligent Systems Group,

The Department of Computer Science

The University of York.

September 1994

‘It is high time we turned to Grammar now,’ said Doctor Cornelius in a loud voice. ‘Will your Royal Highness be pleased to open Pulverulentus Siccus at the fourth page of his *Grammatical Garden or the Arbour of Accidence pleasantlie open’d to Tender Wits?*’

After that it was all nouns and verbs till lunchtime, but I don’t think Caspian learned much.

C. S. Lewis, *Prince Caspian*.

## Abstract

Practical text processing systems need wide covering grammars. When parsing unrestricted language, such grammars often fail to generate all of the sentences that humans would judge to be grammatical. This problem undermines successful parsing of the text and is known as *undergeneration*. There are two main ways of dealing with undergeneration: either by sentence correction, or by grammar correction. This thesis concentrates upon automatic grammar correction (or machine learning of grammar) as a solution to the problem of undergeneration. Broadly speaking, grammar correction approaches can be classified as being either *data-driven*, or *model-based*. Data-driven learners use data-intensive methods to acquire grammar. They typically use grammar formalisms unsuited to the needs of practical text processing and cannot guarantee that the resulting grammar is adequate for subsequent semantic interpretation. That is, data-driven learners acquire grammars that generate strings that humans would judge to be grammatically ill-formed (they *overgenerate*) and fail to assign linguistically plausible parses. Model-based learners are knowledge-intensive and are reliant for success upon the completeness of a *model of grammaticality*. But in practice, the model will be incomplete. Given that in this thesis we deal with undergeneration by learning, we hypothesise that the combined use of data-driven and model-based learning would allow data-driven learning to compensate for model-based learning's incompleteness, whilst model-based learning would compensate for data-driven learning's unsoundness. We describe a system that we have used to test the hypothesis empirically. The system combines data-driven and model-based learning to acquire unification-based grammars that are more suitable for practical text parsing. Using the Spoken English Corpus as data, and by quantitatively measuring undergeneration, overgeneration and parse plausibility, we show that this hypothesis is correct.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Undergeneration . . . . .	3
1.2.1	Definition . . . . .	3
1.2.2	Causes . . . . .	5
1.2.3	Criteria for successful treatment of undergeneration . . . . .	7
1.3	Dealing with undergeneration . . . . .	8
1.4	A novel approach to dealing with undergeneration . . . . .	9
1.4.1	Overview of the approach . . . . .	9
1.4.2	Assumptions . . . . .	10
1.5	Overview of the thesis . . . . .	11
<b>2</b>	<b>Related Work</b>	<b>13</b>
2.1	Introduction . . . . .	13
2.1.1	Sentence correction . . . . .	13
2.1.2	Grammar correction . . . . .	14
2.1.3	Systems reviewed . . . . .	16
2.2	Sentence correction . . . . .	17
2.2.1	The NOMAD System . . . . .	17
2.2.2	Weischedel’s meta-rules . . . . .	19
2.2.3	Parse fitting . . . . .	20
2.2.4	Carbonell et al. . . . .	22
2.2.5	Mellish . . . . .	24
2.3	Grammar correction . . . . .	25

2.3.1	Berwick . . . . .	26
2.3.2	Vanlehn and Ball . . . . .	27
2.3.3	DACS . . . . .	30
2.3.4	The Inside-Outside algorithm . . . . .	30
2.3.5	MIP . . . . .	32
2.4	Incomplete theories and machine learning . . . . .	33
2.4.1	EITHER . . . . .	33
2.4.2	Fawcett . . . . .	34
2.5	Discussion . . . . .	35
<b>3</b>	<b>The Grammar Garden</b>	<b>38</b>
3.1	Introduction . . . . .	38
3.2	System overview . . . . .	38
3.2.1	The set of knowledge sources . . . . .	39
3.2.2	The rule construction mechanism . . . . .	44
3.2.3	The control strategy . . . . .	46
3.3	A worked example . . . . .	51
3.4	Implementation . . . . .	54
3.5	Conclusion . . . . .	56
3.5.1	Meeting the success criteria . . . . .	57
3.5.2	Completeness . . . . .	57
3.5.3	Termination . . . . .	58
3.5.4	Complexity . . . . .	58
<b>4</b>	<b>Model-Based Learning</b>	<b>60</b>
4.1	Introduction . . . . .	60
4.2	Model-based learning in machine learning and in linguistics . . . . .	60
4.3	The initial grammar and lexicon . . . . .	63
4.4	The model . . . . .	67
4.4.1	Linear precedence rules . . . . .	68
4.4.2	Types . . . . .	69
4.4.3	Feature-passing conventions . . . . .	71
4.5	Model-based rule learning . . . . .	72

4.6	Discussion . . . . .	78
<b>5</b>	<b>Data-Driven Learning</b>	<b>80</b>
5.1	Introduction . . . . .	80
5.2	Data-driven learning in machine learning and in linguistics . . . . .	80
5.3	A data-driven grammar learner . . . . .	82
5.4	Examples of data-driven grammar learning . . . . .	86
5.5	Discussion . . . . .	90
<b>6</b>	<b>Evaluation</b>	<b>93</b>
6.1	Introduction . . . . .	93
6.2	Metrics . . . . .	95
6.2.1	Measuring undergeneration . . . . .	95
6.2.2	Measuring overgeneration . . . . .	95
6.2.3	Plausibility . . . . .	96
6.2.4	Comments . . . . .	98
6.3	Experiments . . . . .	98
6.3.1	Undergeneration . . . . .	103
6.3.2	Overgeneration . . . . .	103
6.3.3	Plausibility results . . . . .	104
6.3.4	Convergence results . . . . .	106
6.4	Discussion . . . . .	108
<b>7</b>	<b>Conclusions</b>	<b>110</b>
7.1	Introduction . . . . .	110
7.2	Assumptions . . . . .	111
7.3	Further work . . . . .	114
7.4	General conclusions . . . . .	115
<b>A</b>	<b>The Lexicon</b>	<b>116</b>
<b>B</b>	<b>The Original Grammar</b>	<b>129</b>
<b>C</b>	<b>The Learnt Grammars</b>	<b>134</b>



# List of Figures

1.1	$\mathcal{N}$ and $L(G)$ . . . . .	3
1.2	$\mathcal{N}$ , $C$ , $P$ and $L(G)$ . . . . .	4
2.1	A fitted parse tree. . . . .	22
3.1	An empty chart . . . . .	47
3.2	A chart with lexical edges . . . . .	48
3.3	A chart after proposing edges . . . . .	49
3.4	A chart after extending edges . . . . .	49
3.5	The resulting chart after a failure to parse . . . . .	52
3.6	The chart after seeding with binary super rules . . . . .	52
3.7	The chart after extending a super rule instantiation . . . . .	52
3.8	The chart after extending another super rule instantiation . . . . .	53
3.9	The final parse tree . . . . .	54
6.1	The system's learning curve . . . . .	107
6.2	The increase in sentences generated . . . . .	107
6.3	The learning curve when using more training sentences . . . . .	108



## Acknowledgements

I would like to thank Derek Bridge for supervising my work. Without his attention to detail, this thesis would not exist. In particular, Derek has helped me on the long road to clearer writing.

Thanks to Eric Atwell for supplying the Spoken English Corpus and for being a sparring partner, Ted Briscoe for supplying the SEC Grammar, and John Carroll for technical help with the GDE.

I wish to thank SERC for funding, and this Department for providing computing resources and for allowing me to go to various conferences.

On a less serious note, a hearty roar to the members of the Intelligent Systems Group at York: Robert Dormer, Alan Frisch, Tony Griffiths, and Craig MacNish. How can I sum up three years of coffee, loud tapes, chocolate bars, sympathy, and shouting? Beats me.

Hmm, this is becoming a long cast list. I'll stop by thanking Sr. Amadeus Bulger, IBVM, for helping me in times of love and grief, and finally Amy Davies.

Good God!

## Declarations

I hereby declare that my thesis entitled “Learning Unification-Based Natural Language Grammars” is not the same as that of any that I have submitted for a degree, diploma, or other qualification at any other University or similar institution.

I further declare that no part of my dissertation has already been or is being concurrently submitted for any degree, diploma, or other qualification.

I further state that this research and its results are for the most part my own. Where I have adopted theories or results of others I state this clearly in the text and the bibliography.

Part of this research has been published already [90, 89, 92, 91].

I further confirm that my thesis does not exceed 100,000 words in length.

# Chapter 1

## Introduction

### 1.1 Introduction

The research described in this thesis is an attempt to solve a problem facing practical *text processing systems*, the problem of *undergeneration*. In the approach taken, undergeneration is tackled by *machine learning of grammar*.

Text processing systems typically consist of a *grammar* describing the syntax of a natural language, a *lexicon* describing lexical information about words, a semantic component that is used to construct the meaning of sentences, and a pragmatic component which is used to construct the non-literal meaning of sentences. They also contains a *parser*, which is a program that produces phrases structure trees for sentences in the language defined by the grammar. Text processing systems can be used in a variety of tasks, such as machine translation, text summarising, natural language interfaces to databases, and so on. An example text processing system is the Grammar Development Environment (GDE) [18].

Allowing practical processing of naturally occurring language places certain demands upon the grammar. Perhaps the most important of these demands is that the grammar is of wide-coverage. That is, the grammar should generate all of those grammatical sentences that the text processing system encounters. Unfortunately, no manually-constructed natural language grammar can generate all of the sentences that humans would judge to be grammatical. The GDE is distributed with one of the largest grammars of a natural language (in this case English) and yet fails to generate the uncontroversially well-formed noun phrase (NP) *all abbeyes and*

*an abbot*, as shown by the following trace of the GDE:

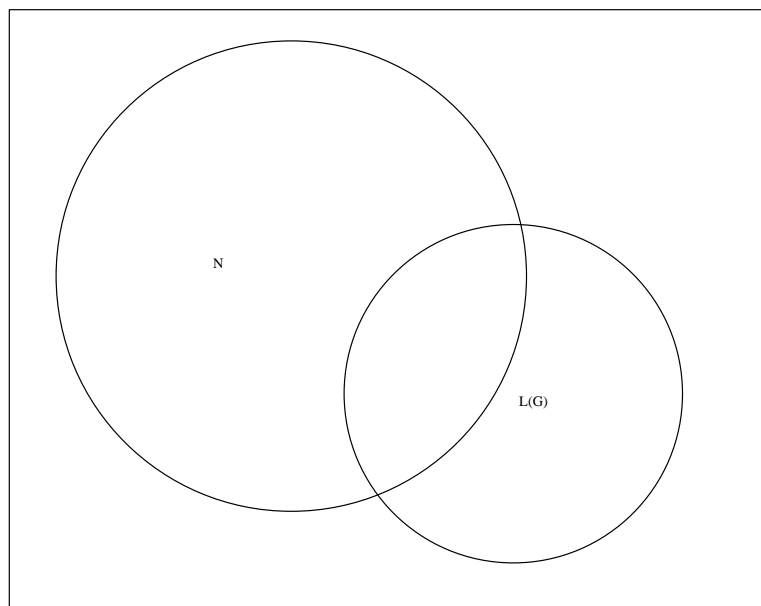
```
69 Parse>> all abbeys and an abbot
28320 msec CPU, 30000 msec elapsed
381 edges generated
No parses
```

This is not due to a lack of lexical information as shown by successful parsing of the NPs *all abbeys and all abbots* and *an abbot and abbey*:

```
68 Parse>> all abbeys and all abbots
50010 msec CPU, 59000 msec elapsed
547 edges generated
1 parse
67 Parse>> an abbey and an abbot
15410 msec CPU, 16000 msec elapsed
238 edges generated
1 parse
```

This failure to generate (in this case) a NP is an instance of *undergeneration*. Undergeneration is a serious problem for natural language processing (NLP) and its treatment is necessary before NLP can be successful. This thesis focuses on overcoming the problem of undergeneration, whilst trying to ensure that the learnt grammar meets the demands made by the host text processing system.

In the next section, undergeneration is defined more carefully. The causes of undergeneration are presented, along with criteria for its successful treatment. After this follows a discussion of two approaches dealing with undergeneration (correcting the string, or correcting the grammar) which will give an indication of how the problem can be tackled. This leads to an overview of the approach presented in this thesis. Finally, a description of what is to come ends this chapter.

Figure 1.1:  $\mathcal{N}$  and  $L(G)$ 

## 1.2 Undergeneration

### 1.2.1 Definition

Let  $\mathcal{N}$  be the set of sentences of some natural language. By ‘natural’ is meant any string of words that a human would judge to be grammatical or acceptable. An acceptable sentence is one that is comprehensible, even if it is not grammatically well-formed. For example, the sentence:<sup>1</sup>

1 *\*Sam chase the ball*

is clearly ungrammatical (there is a subject-verb disagreement), but acceptable (it is obvious what is meant). Let  $G$  be a natural language grammar that we are supplying. The language generated by  $G$  relates to  $\mathcal{N}$  as shown in figure 1.1.

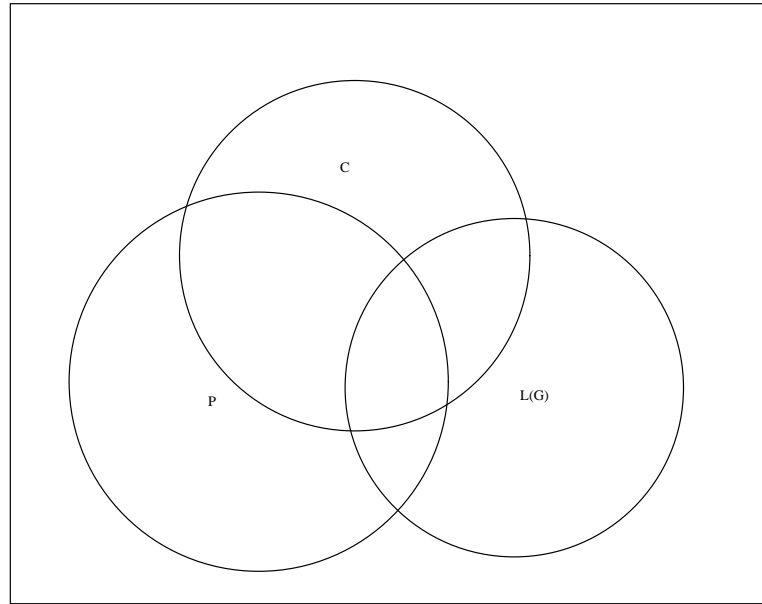
Here, the universe consists of all possible strings that can be formed from  $T^*$ .<sup>2</sup> A first attempt at defining undergeneration would be the set of sentences in  $\mathcal{N} - \mathcal{L}(G)$ <sup>3</sup>

---

<sup>1</sup>Ungrammatical sentences are conventionally marked with an asterisk.

<sup>2</sup> $T^*$  is taken to mean the kleene closure of the set of words in question.

<sup>3</sup>The notation  $L(G)$  means the language generated by the grammar  $G$ .

Figure 1.2:  $\mathcal{N}$ ,  $C$ ,  $P$  and  $L(G)$ 

being not empty. This treats all sentences in  $\mathcal{N}$  equally (they should all be in the language generated by the grammar). Linguists however do not treat natural languages as a monolithic set of sentences. Traditionally, language is demarcated in terms of *competence* and *performance*. Competence describes human knowledge of language and is independent of processes such as memory. Given these two types of sentence, the previous Venn diagram can be refined as shown in figure 1.2. Here,  $C$  is the set of sentences that a human would judge to be grammatical, but possibly unacceptable, and  $P$  is the set of sentences or strings that humans would judge to be either grammatical or ungrammatical, but still acceptable. The previous set of sentences  $\mathcal{N}$  is now the union of  $C$  and  $P$ . There are many interesting regions in this diagram. For example, sentences in  $C$  but not in  $P$  or  $L(G)$ , such as :

**2** *The war the general the president appoints starts ends the world*

will be grammatical, but unacceptable.<sup>4</sup> Such sentences are difficult to process by humans. Any sentence not in  $C$  is ungrammatical and a grammar generating these is said to *overgenerate*. Defining undergeneration now must take into account which

---

<sup>4</sup>This center-embedded example is taken from Johnson-Laird [60, p.271].

region in the Venn diagram  $G$  should expand into. This is controversial. For example, Atwell considers undergeneration to be the existence of sentences in  $P$  but not in  $L(G)$  (personal communication). Instead of Atwell’s definition, undergeneration is defined as being the existence of sentences in  $C$  but not in  $L(G)$ . We believe that the task of a grammar is to generate sentences that, as judged by humans, are ‘grammatical’, and leave ungrammatical strings to a sentence correction approach. Our definition of undergeneration means that a grammar is a theory of competence, and not, if using Atwell’s definition, a theory of whatever strings or sentences the system encounters (or performance). From now on, we shall refer to grammars that try to only generate sentences in  $C$  as a *competence grammar* and grammars that try to generate sentences in  $P$  as a *performance grammar* [117, p.1].

### 1.2.2 Causes

The natural language grammars used in NLP systems undergenerate for two reasons:

- Theoretical. Not all researchers believe that a competence grammar can be constructed, even in theory [104]. Sampson syntactically analysed approximately 10,000 noun phrases (NPs) [104] taken from the Lancaster-Oslo-Bergen (LOB) Corpus [59]. He expected to find most NPs to be of a few types, and the remainder, corresponding to ungrammatical or idiosyncratic NPs, to be of types with far fewer members. Distributions of this kind are known as being *Zipfian* and are common of other aspects of language [133]. For example, function words (“the”, “a”) are far more frequent than other words, whilst words such as “travail” appear only once (for example) in this dissertation. Sampson did not find a Zipfian distribution and concluded that as no such demarcation was found in his experiment between grammaticality and ungrammaticality, no demarcating grammar would be able to account for all of a natural language. Sampson’s conclusion, if correct, profoundly undermines competence grammar construction and so Taylor *et al* re-constructed Sampson’s argument using a wide-covering grammar containing rules of greater generality than those used by Sampson [10, 122]. As Church notes, they found a Zipfian distribution, thereby suggesting that Sampson failed to find an appropriate grammar [27]. Hence, Taylor *et al*’s experiment weakens Sampson’s argument that a compe-

tence grammar can never be constructed that will not undergenerate. It does not totally refute the argument as Taylor *et al*'s wide-covering grammar did still undergenerate.

Assuming competence grammars can be constructed, another theoretical cause of undergeneration is language change. That is, languages evolve over time, and eventually, new constructs will emerge that a NLP systems's grammar, being static, will be unable to generate.

- Logistical. Natural language grammars are large: the descriptive grammar of Quirk *et al.* is over 1500 pages in length [100], the Alvey Tools Grammar [45] contains just under a thousand generalised rules<sup>5</sup>, the CLARE grammar of the CORE Language Engine [1] and the IBM Grammar [112, 6] contain a similar number of rules to the Alvey Tools Grammar. Natural language grammars thus represent a major investment of skilled labour. Some researchers go as far as saying that, in practice, the task of constructing a competence grammar is so large as to be impractical [119, 120, 117]. Atwell, O'Donoghue and Souter comment that [118, p.3]:

Since the syntax of a natural language such as English is extremely complex, large corpora of texts will continue to throw up sentences which are not dealt with adequately by current generative grammars.

Similarly, Souter and O'Donoghue conjecture [120, p.2]:

We are, then, somewhat cautious as to the ultimate value of manually building a large rule-based grammar, as there will always be some new sentences which contain structures not catered for in the grammar, so any parser using such a grammar will hardly be robust.

That is, their conclusion is that the logistical problems involved with manually constructing a wide covering grammar will be so daunting that grammars will always undergenerate. As an example Taylor *et al*'s [122] use of the

---

<sup>5</sup>These rules are *unification-based*. See §3.2.1 for a description of unification-based grammars.



Alvey Natural Language Tools (ANLT) Grammar [45] still failed to account for 3.12% of the 10,000 NPs found in the LOB Corpus. It is difficult to refute the logistical argument, given the fact that, to the author's knowledge, all manually constructed wide-covering competence grammars constructed to date undergenerate.

The theoretical arguments are controversial. If they do not hold, then there is no *inherent* reason why the logistical arguments cannot be overcome by devoting even greater resources to grammar development projects. However, the logistical problems still need to be solved before wide covering grammars can be constructed. This thesis considers how these problems might be overcome by automated support of the grammar engineering process.

### 1.2.3 Criteria for successful treatment of undergeneration

An approach successfully dealing with a grammar  $G$ 's undergeneration should extend  $G$ , giving the grammar  $G'$ , such that  $G'$  satisfies the following criteria:

1. Ideally,  $L(G') = C$ . In practice,  $G$  may generate sentences not in  $C$ , so we have  $L(G') \cap C \supseteq L(G) \cap C$ .
2. Ideally,  $L(G') \cap \overline{C} = \emptyset$ . However, increasing the coverage of  $G$  might lead to overgeneration. In practice,  $(L(G') - C) \subseteq (L(G) - C)$
3. Ideally,  $G'$  should assign the 'correct' set of parses to those sentences that it does generate. In practice,  $G'$  may overgenerate and so the best that that we can hope for is that  $G'$  should assign *plausible* parses to those sentences that it generates.
4.  $G'$  should be in a form suitable for NLP.

$C$  is the set of sentences generated by a competence grammar mentioned in §1.2.1.

The first criterion follows from the definition of undergeneration introduced in the previous section. That is, the extended grammar should generate all the sentences in  $C$ , along with those sentences that the grammar could originally generate.

The second criterion means that  $G'$  should not overgenerate. As previously mentioned, overgeneration is to be avoided. Minimising overgeneration is important

as overgeneration undermines semantic interpretation and produces spurious parses, thereby increasing the chance of the NLP system being swamped.

The third criterion is due to the fact that contemporary theories of semantics and pragmatics rely upon constituents being identified. A grammar that simply recognised a sentence as being within its language, yet failed to assign a plausible parse, makes the task of semantic interpretation harder.

The final criterion follows from demands made by NLP systems upon the formalism used to encode the grammar. These demands include the grammar being declarative, formal, computationally tractable, and explicit. For example, it would be possible to extend the grammar implicitly. Project April has a ‘grammar’ that consists of a set of parse trees and a matching algorithm [46]. As such, the grammar is implicit in the parse trees and the matching algorithm. However, implicit grammars are difficult to reconcile with contemporary theories of semantic interpretation.<sup>6</sup> It is not clear how if the grammar rules are implicit, semantic rules can be paired with corresponding syntactic rules. Furthermore, implicit grammars require ad-hoc parsing algorithms, are of an uncertain coverage and cannot be used to generate sentences. Although this criterion is obvious, not all treatments of undergeneration adhere to it.

### 1.3 Dealing with undergeneration

The problem of undergeneration is a special case of the problem of *robust* text parsing. A robust text parser will be able to deal with *any* sentence encountered in some text. Broadly speaking, there are two ways NLP system designers can achieve robustness:

- Correct the sentence such that the NLP system considers the sentence now to be within  $L(G)$ . This encompasses a set of ad-hoc approaches that are used in applications such as interpreting short, ragged messages and in grammar checking. Undergeneration is reduced, at the expense of not being permanent.

The NLP system will always need the correcting approach in order to deal

---

<sup>6</sup>Contemporary semantic interpretation tends to pair semantic rules with syntactic rules.

with undergeneration. A number of applications using sentence correction are presented in chapter two.

- Correct the grammar  $G$ , giving grammar  $G'$ , such that some sentence  $W$  which is not in  $L(G)$  is within  $L(G')$ . This encompasses a set of machine learning approaches, some of which also appear in chapter two. Unlike sentence correction, grammar correction deals permanently with cases of undergeneration.

Approaches that just use sentence correction might be able to deal with strings in  $P$ , but they erroneously treat sentences in  $C$  as if they were strings in  $P$ . Approaches that just use grammar correction err the other way round. That is, they tend to acquire a performance grammar, and cannot determine when a string in  $P$  is encountered. The optimal approach to the problem of robust text parsing would be one that used sentence correction for strings in  $P$ , and grammar correction for sentences in  $C$ . No system uses both of these approaches, given the complexity of both tasks. The grammar learner presented in this thesis is so designed that it can be used in conjunction with a sentence corrector.

## 1.4 A novel approach to dealing with undergeneration

In this section, an approach to dealing with undergeneration by machine learning is presented. Novel aspects of the learner include:

- The combined use of data-driven and model-based learning to acquire plausible natural language grammars.
- Learning competence grammars, and not performance grammars.
- Using a unification-based formalism.

### 1.4.1 Overview of the approach

When presented with an input string,  $W$ , an attempt is made to parse  $W$  using  $G$ . If this fails, the learning system is invoked. First, the learning system tries to generate rules that, had they been members of  $G$ , would have enabled a derivation sequence for  $W$  to be found. This is done by trying to extend incomplete derivations using

what are called *super rules*. Super rules enable at least one derivation sequence to be found for  $W$ .

Many instantiations of the super rules may be produced by the parse completion process that was described above. Linguistically implausible instantiations must be rejected and this rejection process is interleaved with the parse completion process. Rejection of rules is carried out by model-driven and data-driven learning processes.

If no instantiation is plausible, then the input string  $W$  is deemed ungrammatical. Otherwise, the surviving instantiations of the super rule are linguistically plausible and may be added to  $G$  for future use.

As will become clear from the rest of the thesis, this approach extends a grammar in a manner that meets the success criteria for dealing with undergeneration.

### 1.4.2 Assumptions

The final chapter considers the assumptions made in greater detail. These assumptions include:

- Natural language competence is (at least) strictly context free [97]. This (engineering) assumption is usually made in NLP systems. However, it should be noted that some languages, for example Swiss German, contain constructs that appear not to be context free [114]. These constructs are rare and are usually ignored in most NLP systems [41, p.147].
- Syntax is important for NLP. A position of syntax being distinct from semantics and the other knowledge sources that there might be in a natural language system has been adopted. The reasons for this differentiation include modularity (it is far easier to change the semantic formalism without having to change the syntax if the two are not intimately entwined), and the freedom to select a far wider range of parsing algorithms (for example it is difficult to see how an efficient parser could be used with Small's approach to natural language processing<sup>7</sup> [116]).

---

<sup>7</sup>The syntax of Small's approach is implicit. Efficient parsing algorithms such as the LR family require an explicit grammar to compute the LR-characteristic machine.

- Binary and unary rules are sufficient for plausible syntactic analyses. As most rules in manually written grammars such as the ANLT example are either unary or binary, this is a plausible assumption to make.
- The lexicon is complete. This is the strongest assumption and means that the system assumes that each word encountered is already tagged with its part-of-speech. Given the advent of robust lexical taggers (e. g. [24, 28, 8]), this assumption is commonly made by other workers.<sup>8</sup>
- Competence and not performance grammars are learnt. Most other researchers learning grammar acquire performance and not competence grammars. This is because such researchers use inductive methods, with no model of grammaticality, and so treat any sentence encountered as being grammatical, regardless of how deviant that sentence may be. Granted the fact that overgeneration undermines successful NLP and that performance grammars (by definition) overgenerate, the research reported in this thesis attempts to learn competence grammars. Performance is assumed to be dealt with by rule-based psycholinguistic processes.

## 1.5 Overview of the thesis

There are six other chapters in the thesis:

- Chapter two presents related work: methods that deal with undergeneration by correcting strings into member of  $L(G)$ , methods that overcome undergeneration by correcting the grammar, and finally, related machine learning work on theory (grammar) correction.
- Chapter three presents a novel method of overcoming undergeneration. Firstly the approach is described, secondly a worked example is presented, and thirdly, the implementation of this theory is presented. Finally, properties of the learner are discussed.

---

<sup>8</sup>Work on automating lexical acquisition (for example [67, 103, 33, 54]) further supports the complete lexicon assumption.

- Chapter four expands upon the concept of model-driven (deductive) learning, and then goes on to both show how model-driven learning can help to learn grammatical rules and to show the failings of model-driven learning.
- Chapter five presents data-driven (inductive) learning, which complements model-driven learning, and then goes on to both show how data-driven learning can help to learn grammatical rules and also to show failings of data-driven learning.
- Chapter six links the previous three chapters by evaluating the entire system. Evaluation shows the contribution that model-driven and data-driven learning make upon the quality of the grammars learnt. Principally, the chapter considers the hypothesis that using both learning styles is better than using either learning style in isolation. The results of the evaluation show that this hypothesis is correct.
- Chapter seven, the final chapter, summarises the research, reconsiders the assumptions made, shows ways of extending this work, and ends by making some general conclusions.

## Chapter 2

# Related Work

### 2.1 Introduction

As mentioned in chapter one, approaches deal with undergeneration either by correcting the sentence, or by correcting the grammar. Both of these approaches will now be considered in more detail.

#### 2.1.1 Sentence correction

Correction is defined as mapping a string not in  $L(G)$  into one or more sentences that are in  $L(G)$ . There are a variety of ways of implementing this relation, some of which relate to formal languages and not necessarily to natural languages.

The simplest method would be to skip over the point of difficulty in the hope that the parser could be restarted. This is known appropriately enough as *panic mode* [43]. To be of any use, the parser needs to have an idea of what constitutes a point of synchronisation in the sentence. In a block structured language such as Pascal, such a point might be a *begin* statement. In the natural language context, this might be a phrasal head. As may be seen, the synchronisation point needs to be specified. Panic mode can be seen as a (limited) form of correction in that part of the sentence is deleted (ignored) [47]. However, the deletion may also remove an arbitrary amount of the sentence. Consider the example of:

**3** *\*The ants eats everything in the kitchen.*

Here we have a subject-verb disagreement. A left to right parser would consider the

word *eats* and fail. Panicking might involve trying to look for a verb that agreed further in sentence 3. Such a search would exhaust the sentence here, but might work for a restarted sentence. A restarted sentence is one in which the speaker can be seen to change her plan (as realised in the sentence) whilst in the process of constructing the sentence in question, for example:

4 \**The ants eats everything eat everything in the kitchen ...*

Correction can also change the sentence. The idea here is that a sentence not in the language generated by the grammar is a distorted version of a grammatical sentence and if the underlying content of the grammatical sentence is predictable then there can be an informed mapping from the former to the latter. This is the optimal method for dealing with undergeneration with respect to corrective ability, though computationally difficult to achieve. For example, the sentence in *P*

5 \**Sam chase the cat*

could be corrected into the sentence in *C*:

6 *Sam chases the cat*

Correcting sentence 5 into 6 assumes that the verb does not agree with the subject (and not vice versa). In general, there will be many possible corrections that can be made.

### 2.1.2 Grammar correction

Correction of the grammar can be defined as an attempt to extend the grammar  $G$ , giving grammar  $G'$ , such that some sentence not in  $L(G)$  is within  $L(G')$  and  $L(G) \subset L(G')$ . Grammar correction is an example of *machine learning*.

Carbonell and Langley comment that attempting to define *machine learning* formally is always open to controversy [68]. Nevertheless, the learning task can be stated as: given one or more instances of a class, find a description that predicts which class a new instance is a member of. In the context of grammar learning, learning would consist of creating a description that (amongst other things) predicts whether a sentence is grammatical or ungrammatical.



Approaches to machine learning of grammar can be grouped broadly into whether they are *inductive* (or *data-driven*) or *deductive* (or *model-driven* or *explanation-based*).

In inductive learning the task is to construct a description that covers a set of positive examples and does not cover any of the set of negative examples. There are many implementations of this scheme, including the use of version spaces [83] (in which a set of possible descriptions is maintained), decision trees [98] (a series of questions, the answers to which characterises the concept) and the early example of Winston's Arch, which refined a single concept of the arch from near misses [129]. Inductive learning is important in many problem solving applications, for example in speech recognition and in learning a language. To learn the classification, a set of features needs to be defined that demarcate the space of possible examples. So, for the speech example, we may have features that correspond to those found in phonology (segment, nasal, sonorant, etc.). Each instance then becomes a vector of such features. Note that inductive learning is not sound: the learnt concept that has evolved may at some later stage need to be revised, given a counter-example. The usual example is that of trying to devise a law that characterises the colour of sheep. By seeing a flock of white sheep on the hill, it could be concluded that sheep are white, but the black sheep that is out of sight will cause this theory to be revised.

*Deductive learning* is less data-driven as only a few examples of the concept to be acquired are needed [30]. The idea is to determine why a given example is an example. This is achieved with domain-specific knowledge. Once the explanation is constructed, generalisation can then take place. Rich gives an example of a fork in chess [102, p.472]. The learning program would explain why this is a bad position and then try to generalise this explanation by discarding unimportant aspects of the explanation. So, from an instance of a particular fork, the learner would be able to apply this knowledge to learn about forks in general. Constructing an explanation is similar to constructing a proof of the example being deducible from the domain-theory and hence deductive learning is both sound and also can be viewed as a search over the domain theory. Note that if the domain theory is incomplete, the deductive learner will be unable to learn any of the examples that cannot be proved. Again,

there are many examples of deductive learning systems, as surveyed by Ellman [30] and Mitchell, Keller and Kedar-Cabelli [82].

In sum, sentence correction can deal with undergeneration, but the solution is temporary, and arguably, does not address the issue of undergeneration as being due to a deficient grammar, and not being due to the string being ungrammatical. Grammar correction also deals with undergeneration, but permanently, and directly. For the sake of completeness however, this chapter considers both approaches, as outlined in the next section.

### 2.1.3 Systems reviewed

The rest of this chapter presents both sentence correction and also grammar correction approaches dealing with undergeneration. Machine learning researchers also deal with undergeneration under the guise of the *incomplete theory problem* and hence it is useful to consider such related work. The chapter ends with a discussion of how well undergeneration is dealt with by the previously mentioned approaches.

The systems reviewed in this chapter can be broadly classified as follows:

System	Boundary recognition	Corrects	Learns	Plausible	Explicit
NOMAD	No	Yes	No	?	Yes
Weischedel	Yes	Yes	No	No	Yes
Parse fitting	No	Yes	No	No	Yes
Carbonell	No	Yes	No	?	Yes
Mellish	Yes	Yes	No	?	Yes
Berwick	Yes	No	Yes	Yes	Yes
Vanlehn and Ball	No	No	Yes	No	Yes
DACS	No	No	Yes	No	Yes
Inside-Outside algorithm	No	No	Yes	No	Yes
MIP	No	No	Yes	No	No

The first five systems deal with undergeneration by sentence correction, whilst the other five deal with undergeneration by grammar learning.

In the table, *boundary recognition* means that the approach recognises the difference between performance, competence and word salad. *Corrects* means that the system changes sentences not in the language generated by the grammar into one

or more sentences that are generated by the grammar. *Learn* means that the system corrects the grammar. *Plausible* is meant to suggest that the system produces linguistically plausible parse trees for sentences in the language generated by the grammar. Plausibility is difficult to define precisely [6, p.5], but roughly stated, can be thought of as saying that constituents are identified correctly in the parse. Identification is stronger than simply bracketing a sentence, which is a common criterion for parse plausibility [6, 49]. *Explicit* is meant to say that the NLP system's concept of grammaticality is expressed as a distinct, symbolic grammar.

## 2.2 Sentence correction

Here, a variety of sentence correction approaches are presented. As previously mentioned, there is little over-arching theory of correction and hence all of these systems adopt a variety of strategies.

### 2.2.1 The NOMAD System

Granger's NOMAD system converts extremely ragged messages into a database readable form [44]. The messages are from a restricted domain of Naval ship-to-shore dialogue, for example:

**7** *Locked on open fired destroyed*

This message lacks punctuation, and lacks subjects and objects for the verb phrases. NOMAD would produce for the above message:

**8** *We aimed at an unknown object. We fired at the object. The object was destroyed.*

The system attempts to process texts in a left to right manner, and uses scripts [108] to give a predictive element. Each word that is processed suggests new expectations. These expectations contribute to the meaning of the text, and when they are not met, result in 'surface-text' alerts. The alerts can be due to unknown words, missing subjects or objects, missing clause boundaries, ambiguous word usage, or a lack of tense agreement.

The algorithm that NOMAD uses is:

```

parse the message
if a blockage occurs
    set an alert flag
    try to continue to parse
if the text cannot be fully interpreted
    use the failure in interpretation to resolve the alert flag

```

NOMAD encodes linguistic knowledge in terms of word-level routines (similar to Small's Word Experts [116]) and relies on user interaction to resolve potential solutions.

As Granger himself notes, NOMAD is difficult to extend, due to the use of word-level routines. The use of scripts as the predictive element is worrying, given their well known problems (i.e. difficulty in matching, and lack of flexibility). NOMAD uses a failure to understand as a guide to locating the cause of the grammatical error, a form of blame assignment. To do this well, the semantics component must be correct (as a failure here will limit any attempt at dealing with undergeneration) and so the approach is only as good as the expectation mechanism.

If the parser cannot make sense of a fragment, yet the text is still comprehensible, then according to Granger, the fragment will be ignored. For example, faced with

**9** *Toby ate the lobster and Jane crab.*

NOMAD, if armed with a restaurant script, might generate:

**10** *Toby ate the lobster.*

Here, an expectation has been found (a person eating in a restaurant) and so the other material would be incorrectly ignored.

There is no description of the effectiveness of the approach, other than the size of the texts being up to 17 sentences in length. It is suspected that the approach will not scale-up easily to that of an open domain, where the connection between a failure to understand and a solution to undergeneration may not be so obvious. This is an example in natural language processing of a problem that is tackled through constraining the domain, and thus making the solution of less general value.

### 2.2.2 Weischedel's meta-rules

Weischedel advocates relaxing grammatical constraints when dealing with under-generation [127]. Relaxation can be considered as a form of sentence correction. For example, if the grammar lacked rules dealing with transitive verbs and had rules dealing with (say) ditransitive verbs, encountering a transitive verb would result in the system treating the transitive verb as being ditransitive, albeit with an NP implicitly introduced into the sentence. Such relaxation is carried-out by the application of meta-rules to either the productions of the grammar, or to the parsing configuration. Meta-rules are said to correspond to different types of error. Each meta-rule is of the form:

$$C_1 C_2 \dots C_n \rightarrow A_1 A_2 \dots A_n$$

where  $C_i$  is the  $i^{th}$  condition that has to be met, and  $A_j$  is the  $j^{th}$  action that is made if all of the conditions on the left hand side of the rule are met.

An example meta-rule that deals with a failure for the subject and the verb to agree is:

```
(failed-test? (subject-verb-agree? ?x ?y))
--> (new-configuration
      (failed-constraint (subject-verb-agree ?x ?y)
        (substitute-in-arc (subject-verb-agree ?x ?y) T)))
```

That is, the test `(subject-verb-agree ?x ?y)` is replaced by the Lisp atom `T`, which always evaluates to true. Here the parser configuration is being changed to allow processing to continue. In this case, we are dealing with an ATN formalism<sup>1</sup> but the author of the paper suggests that the approach is applicable to any parser that is capable of being expressed in terms of a configuration, an obvious example being an LR parser.

The approach is to apply a meta-rule when parsing is blocked. The meta-rule is said to diagnose the error, relax the violated rule, add a ‘deviancy note’ and allow processing to continue.

---

<sup>1</sup>An *ATN* (Augmented Transition Network) is one of many implementations of a grammar and parser [130].

The errors that are said to be dealt with include: failed grammatical tests, word confusions, spelling errors, unknown words, restarts and contextual ellipses.

The true nature of an (apparent) error is not always obvious. Consider the sentence:

**11** *Toby kicked the red ball and Jane the blue one.*

If the system had a simple view of coordination, with rules of the form  $\alpha \rightarrow \alpha \text{ conj } \alpha$ , where  $\alpha$  is a string of terminals or nonterminals, and *conj* is a nonterminal symbol for a coordinating lexical category, then the above sentence might be either a failure to coordinate sentences, or may be a case of ellipsis. Weischedel ranks such hypotheses by the amount of material that has been processed and by a partial ordering on the meta-rules themselves. The details of the ordering are not given in the paper. Of course, it is not clear what this ordering should be. The hypothesis that accounts for most of the sentence is preferred over those that account for less.

If the parser cannot continue, due to the meta-rules not accounting for the ‘error’, then processing is abandoned. The parser will still allow undergeneration to take place therefore, and this will be as a result of oversights in the coverage of error classes by the meta-rules and inadequacies of the approach.

Using meta-rules is interesting, but the performance of the scheme is bounded by the ability to account for errors. Also, as Weischedel notes, “Significant effort is required of the grammar writer to devise the condition-action pairs.” [126, p.95].

### 2.2.3 Parse fitting

EPISTLE is an attempt at providing a grammatical and stylistic critique of technical English texts [58]. The parser uses an augmented phrase structure grammar along with a lexicon of 130,000 entries. Parsing is divided into three parts: simple parsing using the (‘core’) grammar that defines the uncontroversially grammatical structures, a set of procedures to handle ambiguity, and finally the fitting procedure. The core grammar consists of about 300 rules, and these (are reported to) account for 70% of all sentences that are to be analysed. Ambiguity is resolved by a metric that ranks alternative parses.

In cases where the parser fails to find an analysis for a sentence, the fitting

procedure tries to fit the parse fragments (in a global manner) into a tree whose root is the start symbol of the grammar. The algorithm for fitting is in two stages. The first stage attempts to find a head constituent<sup>2</sup> from an ordered list of candidates, the candidates being the fragments that are built by the parser. The second stage then tries to ‘glue’ the remaining pre and post positional fragments to the head constituent, such that all of the material is accounted for.

The head constituents are searched for in decreasing order of preference: VPs with tense and subject, VPs with tense and no subject, phrases without verbs (NPs, PPs), non-finite VPs and finally ‘others’. If more than one candidate is found then the candidate that accounts for more of the material is selected.

Should the head constituent not cover the entire sentence, then the remaining constituents are added on to either side of the head constituent, with the following order of preference: non-VP fragments, untensed VPs and finally tensed VPs. The overall effect of fitting is to select the largest chunk of sentence-like material and to attach left over chunks in some reasonable manner. Both the set of head constituents and ‘fillers’ are considered to be principles of syntactic well-formedness.

Here is an example from the paper. Consider the following input string:

**12** *Example: Your percentage of \$ 250.00 is \$ 187.50.*

This string is considered to be a sentence inasmuch as it begins with a capital and ends with a full stop. However, the core grammar does not consider it to be grammatical and so will fail to assign a complete analysis. The set of parse fragments constitute the basis of subsequent repair.

The first step is to look for a head constituent amongst the set of fragments. There is an ordering of candidates for being a head constituent, and the algorithm initially looks for VPs with tense and subject. Candidates include:

\$250.00 is

percentage of \$250.00 is

\$250.00 is \$187.50

Your percentage of \$250.00 is \$ 187.50

---

<sup>2</sup>The head of a rule is a category in the rule’s right hand side that characterises the rule. For example, the head of a noun phrase is a noun. The head of a constituent is therefore the head of a rule used in a parse tree. See §3.2.2 for a more detailed discussion of heads.

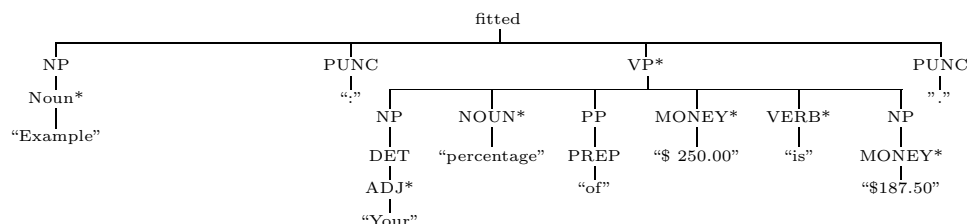


Figure 2.1: A fitted parse tree.

The final candidate accounts for most of the material of the sentence and is therefore chosen. Note that if no tensed VP with subject was found, the algorithm would then look for a tensed VP with no subject, and so on.

Fitting then continues by adding material that is to be found to the left and to the right of the chosen constituent. Here, this results in “Example:” being found prior to the head constituent and the full stop posterior to the constituent. These are therefore added to the fitted parse tree, resulting in a tree that spans the entire sentence, as shown in figure 2.2.3. The starred nodes are nodes that are fitted together.

Parse fitting will always produce a tree that spans any sentence, no matter how badly formed that sentence might be. At the worst case, the tree will consist of a root node immediately dominating the lexical items. Success follows from parse fitting’s liberal definition of a well-formed tree: for a tree to be well-formed it does not have to have (say) an S node dominating the entire sentence; all that is required is that the fragments be joined up. This therefore tries to make sense of what has been tried and does little to correct the sentence or the parse tree. There is an idea of what a sentence should be composed of in the ordering of material to look for when searching for candidates. This idea of sententiality is however a little mysterious in that it is not obvious why the authors have chosen this particular set, and why this particular ordering. In their paper, they do not motivate their choice at all.

## 2.2.4 Carbonell et al.

Researchers at Carnegie Mellon have developed a series of robust parsers around a common theme of using case frame instantiation in a restricted domain. A case



frame is a structure that makes explicit the semantic roles of (for example) subjects and objects in sentences. These parsers include: FlexP [51], CASPAR [52], DYPAR [125], DYPAR II [56] and Multipar [31]. As these are all broadly similar, CASPAR and DYPAR II shall be concentrated upon (these are reasonably well documented).

CASPAR tries to deal with simple imperative VPs in a robust manner. For example,

**13** *Cancel math 247*

**14** *Enrol Jim Campbell in English 324*

These can be seen as examples of case constructions in that the verb is the central concept and the attendant NPs are the arguments. Parsing is designed to exploit this restricted syntax. The algorithm is simply to parse from left to right, applying patterns corresponding to the verb phrase, and if a match is found, look for case fillers.

This approach will usually produce a command and a (possibly) incomplete set of arguments, and is insensitive to extraneous input. Arguably, success depends upon the ability of the case-frame matching process and any unmatched material that is lost could conceivably be relevant.

DYPAR II again uses case frames and tries to combine multiple parsing strategies within a single framework. These strategies include pattern matching, semantic grammars and syntactic transformations. This parser is used in XCALIBUR [56], which allows natural language access to the XSEL expert system. It deals with spelling corrections, ignores garbled or spurious phrases in otherwise acceptable material, recognises constituents when they occur in unexpected order and has a simple ellipsis resolution mechanism. The robustness again stems from the simple minded parsing scheme, which is similar to CASPAR's approach. What is new is the use of XCALIBUR's discourse structure to resolve unfilled case fillers. When XCALIBUR generates a query to the user, there can (within this small domain) be only a limited set of felicitous responses by the user. This set of responses suggest the particular case frames to look for. Discourse is considered to be stack based. If a case frame is only partially filled, then the first (completed) case frame that matches

the current (incomplete) frame is used to supply the missing material. Expectation is also used to constrain the search when dealing with spelling corrections. For example, the sentence:

**15** *\*Add a dual prot disk*

is corrected to

**16** *Add a dual port disk*

as a disc descriptor is expected [56, p.4]. This is a reduction over the 13 possibilities that a spelling program suggested.

Parsing here may be robust, but this is by no means a solution to undergeneration. If the domain of discourse was not so highly specified then trying to skim the sentence in the manner used here would fail. This is simply because the unspecified material of the sentence would be ignored, thus treating undergeneration by pretending it does not exist.

### 2.2.5 Mellish

Mellish presents an approach that tries to ‘fit’ parse fragments into a tree dominated by the start symbol of the grammar [79]. However, unlike parse fitting (§2.2.3), the fragment consolidation process is carried out by relaxing grammar rules in a manner similar to the application of meta-rules (§2.2.2). That is, correction is defined not in terms of linguistic notions such as headness, but in terms of possible errors such as omitted words in a sentence. Mellish’s approach also differs from either meta-rules or parse fitting in that he uses heuristic search to decide which of the many possible parses the parser should choose at any one stage. For example, in the ungrammatical sentence:

**17** *\*The gardener collected manure if the autumn*

the word *if* should be a preposition. Hence, a rule such as  $NP \rightarrow P NP$ , necessary to complete the parse, cannot be applied. Relaxing this rule both treats the word *if* as a preposition and also allows the parse to be completed.

Mellish reports that his approach can deal with single errors, but that due to the large size of the search space, may not scale-up to dealing with either multiple

errors or to using a large grammar. Because Mellish relaxes the grammar and hence presupposes that the necessary grammar is there in the first place, any case of undergeneration will undermine this solution to the problem of correcting errors.

## 2.3 Grammar correction

There are a number of results outlining the conditions for successful grammar learning. Learning successfully means that the system eventually acquires a grammar that meets the success criteria outlined in chapter one. That is, the acquired grammar should not undergenerate, overgenerate, or assign implausible parses to those sentences that it does generate. The system is then said to have *identified in the limit the language* [42]. Gold showed that context free languages can be identified in the limit if the system has an *informant*. An informant is a knowledge source that can be called upon by the system when determining how to revise the grammar. Example informants include, but are not limited to, people (the system would query the user), some function which could be called upon to present the learner with sentences of ever increasing syntactic complexity (ordering the text), negative information, or a model of grammaticality (the system would use the model to prove a sentence's derivation sequence). If the learner does not have an informant, or the informant makes mistakes, context free languages cannot be identified in the limit. Pinker notes that in these two cases, the success criterion for language identification must be relaxed [95]. That is, context free languages will not be identified in the limit and the learnt grammar will only be an approximation of the (hypothetical) grammar that has generated the context free language being identified. Learning systems surveyed in this chapter use all of these versions of an informant. Clearly, not all of these version are equally suitable for all NLP tasks. For example, non-interactive systems cannot query the user and interactive systems operating in real time cannot order the text. However, an informant implemented as a domain theory is suitable for all NLP applications, including text parsing.

The chapter now goes on to present a series of grammar learners.

### 2.3.1 Berwick

Robert Berwick bases his grammar acquisition work on Chomskyan ideas of Universal Grammar (which uses a model-based informant) and on Marcus' Parsifal parser [77] [4], which is (generally held to be) deterministic. Parsifal consists of two data structures: a pushdown stack and a three cell lookahead buffer. The first cell may be filled with either a word or a phrasal category and is the item under consideration when parsing.

Acquisition is simple. If the parser cannot continue then at the point of failure an attempt is made to construct a new grammar rule to allow parsing to continue. If parsing cannot continue with the addition of this rule then the sentence is rejected.

Berwick starts with no grammar.

Upon encountering a case of undergeneration, four possible actions are tried (this is therefore a generate and test method):

- Attach
- Switch
- Insert lexical item
- Insert trace

*Attach* takes the item in the left-most cell and attaches this to the node currently on top of the stack. This therefore corresponds to a late closure. The action is constrained by  $\overline{X}$  Syntax.<sup>3</sup> If this rule fails, then *switch* interchanges the contents of the left-most cell with the cell next to it. This rule succeeds only if the rule following this new rule itself succeeds.<sup>4</sup> The next action is to *insert a lexical item*. This adds into the surface structure a specified closed class word, for example 'you' which may have been deleted in the surface structure (p.146). The final option is to add a *trace*. Note that if all of these potential actions fails to allow parsing to continue then the sentence is rejected. The actions are ordered as shown in the

---

<sup>3</sup>See §3.2.2 for an explanation of  $\overline{X}$  Syntax.

<sup>4</sup>The reasoning behind this seems to be that if a switch is needed, then the sentence being parsed is in reality a transformation of some other syntactic structure.

above list and Berwick claims that this ordering results in the narrowest grammar being acquired (p.113).

Sentence order presentation matters in that the acquisition procedure depends upon an assumption of one rule per error (p.109). Therefore, a (relatively) complicated sentence will contain grammatical structures which need to be learnt from exposure to a simpler sentence. Berwick's approach, for example, needs to learn about prepositional phrases before it can learn about preposed prepositional phrases. There is a question of how local this *finite error condition* (Berwick's terminology) is to be. Berwick is forced into his position by the choice of Marcus's Parsifal, with its narrow idea of sentential context. For long-distance constructs such as gaps, the top of the stack may not relate to the part introducing the gap and so gapping cannot be learnt.

Berwick himself admits that his approach cannot learn coordination (p.38) and will not learn ambiguity (given the strong determinism of the Marcus Parser). There is no explicit analysis of the performance of the system, for example how the system fails, or how plausible the parses produced are. The ordering requirement (of the sentences) means that the parser will have to continually reparses a training set, if a text is being used.

### 2.3.2 Vanlehn and Ball

Vanlehn and Ball present a *version space* approach to learning grammars [124]. A version space is a set of all generalizations consistent with a given set of instances. In the context of learning language this means that for a set of sentences, the version space is a set of grammars that covers the set of well-formed sentences but excludes the set of ill-formed sentences. These generalisations can be given a partial ordering of generality, although this cannot be done for the general case of context free grammars.<sup>5</sup>

It is a well known result that for any language  $\ell$  there is an infinite number of weakly equivalent grammars. This means that the version space for context free

---

<sup>5</sup>The test to see if the language generated by a CFG  $A$  contains the language generated by a CFG  $B$  is undecidable [53]. Version spaces can be used to learn context free grammars if the lattice is not ordered by language inclusion.

grammars is infinite. To avoid this problem, Vanlehn and Ball restrict the grammars by not allowing rules of the form  $A \rightarrow \lambda$ ,  $A \rightarrow B$ , where  $\lambda$  is the empty string and  $A$  and  $B$  are non-terminals, and disallowing useless non-terminals.<sup>6</sup> Furthermore, they introduce the idea of a *reduced grammar*. A reduced grammar is one that given a set of sentences  $P$  cannot be reduced in size by removing a production and still parse  $P$ .

Given these conditions, the version space is said to be finite.

The details of the algorithm are best illustrated by an example adapted from the paper (p.63) in which a command language is acquired. Sentences and non-sentences of this language include:

**18** *delete all*

**19** *\*all delete*

**20** *delete it*

Suppose the first of these sentences was presented. This would result in a set of grammars being produced, the form of the productions being limited to a simple format which allows all possible rule permutations that cover the sentence to be constructed. These grammars might be:

Grammar	Rules
G1	$S \rightarrow \text{delete all}$
G2	$S \rightarrow \text{delete } S, S \rightarrow \text{all}$
G3	$S \rightarrow S \text{ all}, S \rightarrow \text{delete}$
G4	$S \rightarrow S S, S \rightarrow \text{delete}, S \rightarrow \text{all}$

The most specific grammar is G1 and the most general is G4.

If the non-sentence 19 was presented next then the algorithm would try to exclude this negative example from the languages generated by the grammars. This sentence is only generated by G4 and so G4 is ‘split’ in an attempt at exclusion. Splitting must generate a grammar that both generates the positive sentences and

---

<sup>6</sup>A *useless* non-terminal is one that cannot be reached from the start symbol in a derivation, or if reached, does not match the right hand side of any production in the grammar.

also fails to generate the negative sentences, and is achieved by making simple changes to the grammar. The splitting of G4 results in three extra grammars and of these, one still generates a negative sentence and so this grammar is further split into two grammars. One grammar generates a negative sentence and is rejected. The other grammar thus generated is ‘covered’ by an existing grammar and is abandoned.<sup>7</sup>

The two new grammars that are produced by the splitting of G4 are:

Grammar	Rules
G5	$S \rightarrow S A, S \rightarrow \text{delete}, A \rightarrow \text{all}$
G6	$S \rightarrow A S, A \rightarrow \text{delete}, S \rightarrow \text{all}$

This algorithm suffers from a combinatorial explosion evident in the splitting of grammars. For the learning of natural language grammars, it is unclear if this work can be directly used. Firstly natural language grammars seem to violate the restrictions placed upon grammars here (they have unit productions and productions introducing gaps) and so the version space is infinite. A partial ordering is therefore not possible in the general case. Secondly, the result of this particular form of induction is a space of grammars and so the question arises as to which of the grammars is to be used. The most specific grammar is in effect an enumeration of the well-formed sentences of the training set and as such is useless. The most general grammar may overgenerate too much to be useful. Thirdly, to use this approach requires a set of negative sentences. Whilst such a set could be constructed, the need for this appears to be unnatural. What is really required is a method of acquiring the syntactic structure directly from the sentence, and not as a composite from the ill and well-formed sentences. Fourthly, the training set of sentences needs to be kept in order to evaluate new grammars and this set could be arbitrarily large if an extensive grammar of natural language is to be induced. Clearly this is unsatisfactory as one can imagine that the training set would be rather large. An advantage of using version spaces is that it is clear how far the learner has to go to before halting. The search space is indicated by the number of grammars in the lattice and when this contains only a single grammar, the learner can halt.

---

<sup>7</sup>The paper does not give the details of these rejected grammars.

### 2.3.3 DACS

DACS (‘Data driven ACquisition of Syntactic knowledge’) [85] is an inductive, symbolic system that infers HPSG-style grammars [96]. Prior to parsing, the words of the strings of the corpus are assigned their parts of speech (this is known as *tagging*) and then the strings are sorted by length. Sorting the corpus is an approximation of an informant. Once the corpus is sorted, the system then selects the first  $k$  sentences and attempts to parse these. For cases of undergeneration, DACS first generates all possible substrings and then, using heuristic pruning, joins these substrings together. The number of joining operations (a join is called confusingly a ‘gap’) is noted for these sentences and the new rules corresponding to the tree with the lowest number of joining operations are added to the grammar. The corpus is then re-parsed and the process continues until the corpus is generated by the grammar. There is no attempt to learn recursive rules or to impose linguistic plausibility constraints upon what can and cannot be used to form the left hand side of a rule. DACS has no concept of headedness and so learns rules such as  $NP \rightarrow Adj Adv$ . Rules are learnt in a monotonic manner and so bad rules are never rejected. Because the system only joins substrings together, DACS cannot learn ambiguous attachments. DACS obeys the *finite error condition*.

This particular system is interesting because of its use of a parsimonious (unification-based) formalism and because it uses an ordering informant.

### 2.3.4 The Inside-Outside algorithm

Baker’s Inside-Outside algorithm is a popular way of inducing a stochastic context free grammar directly from a text [3]. The algorithm has been used by workers at IBM [112, 6], at Cambridge University [69, 12, 132] and at Brown University [15]. Prior to training, the set of all possible rules (modulo some length of the number of symbols in the right hand side) are given initial probabilities. Then these values are iteratively re-estimated, in a manner similar to training Hidden Markov Models, until convergence is attained. There is no guarantee that the algorithm will converge towards a global optimum: the algorithm hill climbs and hence is not admissible [86]. This lack of convergence is due to induction’s unsoundness in that this particular



method picks a solution that is optimal in a local sense, but not in a global sense. For example, Carroll and Charniak, using the Inside-Outside algorithm to estimate rules for a generated corpus, assigned random initial values to all the rules in the grammar [15]. They reported that for each of 300 different random starting points, a different local minimum was reached and that none of the grammars corresponding to these local minima was correct, pointing out the need for selecting sensible initial values.

Imposing constraints upon the space of rules is one way of helping the Inside-Outside algorithm converge upon a plausible grammar. This insight (in the context of learning natural language grammars) has been exploited both by Briscoe and Waegner [12] and by Carroll and Charniak [15].

Briscoe and Waegner place three restrictions upon rules. Firstly, they restrict rules to be in Chomsky Normal Form (CNF). A grammar in CNF has rules of the form  $A \rightarrow a$  and  $A \rightarrow B C$ , where  $A, B, C$  are non-terminals and  $a$  is a terminal symbol. This places a bound upon the number of possible rules in the grammar. Secondly, a headedness constraint is imposed. This constraint only licenses rules whose left hand side (LHS) is a *projection* of the rule's head category. A projection (roughly speaking) relates the phrase as a whole to the rule's head. Note that this constraint is a little draconian in that some rules, for example those to do with possessives, arguably do not project the head. The third constraint forces rules whose right hand side (RHS) immediately dominates lexical items to have a second category in the RHS that is nominal or adverbial. This constraint seems less well justified than the other two. Briscoe and Waegner compile out the feature-based formalism<sup>8</sup> used into a CNF grammar. The resulting grammar contains 3786 rules and parses about 75% of the Spoken English Corpus. The grammar also hugely overgenerates. The following sentence taken from the same corpus:

**21** *Next week a delegation of nine Protestant ministers from Argentina visits the Autumn assembly of the British Council of Churches.*

receives 2186624992778031036 parses. The parse ranked as most likely, however, is close to the desired parse for such a sentence. Note that there is no rule retrac-

---

<sup>8</sup>A *feature-based formalism* replaces the atomic categories of a PSG with complex categories consisting of sets of *feature-value pairs*. See §3.2.1 for a full description of such a formalism.

tion: the constraints are presumed to be complete. Also, to be successful, such an approach is reliant upon the parse selection mechanism.

Charniak and Carroll constrain the Inside-Outside algorithm by firstly using a dependency grammar (which places similar constraints upon the space of possible rules as does a CNF grammar) with a length bound upon the number of symbols in a rule's RHS and secondly by removing rules from the grammar that have a probability lower than some threshold. Charniak and Carroll do not evaluate their system. They do present the learnt grammar, which appears to be plausible from a casual inspection, but also is extremely small, and thus presumably continues to undergenerate.

Apart from problems of convergence, the Inside-Outside algorithm has a computational complexity of  $O(n^3)$ , in terms of the number of non-terminals in the grammar. Briscoe and Waegner comment that this complexity means that the algorithm cannot be used practically to estimate a wide covering grammar with many non-terminals such as the Alvey Tools Grammar [12]. Lari and Young also note in order to achieve convergence, the algorithm needs more symbols in the grammar than is strictly necessary, thereby exacerbating this inefficiency [69]. The algorithm is also unlikely to learn a grammar that is linguistically plausible, given the vast number of competing, linguistically implausible grammars that could also be induced.

### 2.3.5 MIP

Brill, Magerman, Marcus and Santorini have developed an inductive, stochastic system called the *Mutual Information Parser* (MIP) [9]. MIP, unlike the Inside-Outside algorithm, does not use a conventional grammar at all: the system locates constituency by using a *distributional analysis* and hence is radically non-symbolic. Sequences of word tags are analysed and the hypothesis is that certain subsequences occur sufficiently frequently to be identified as being constituents. MIP uses *mutual information* statistics between tag sequences of a certain length (also called *n-grams*). Mutual information is a measure of the interdependence between these *n-grams*. Parsing using mutual information is a search for a bracketed structure that maximises the partitioning into *n-grams* of the sentence in question. Parser training is done by counting frequencies of tags pairs found in a tagged version of

the Brown Corpus [37]. On testing, MIP made about 2 errors per sentence for sentences of less than 15 words and between 5 and 6 errors for sentences between 16 and 30 words in length. A major problem with MIP is that distributional analysis does not produce labelled parses, which are necessary in order to carry out semantic interpretation. Leeds University's *Constituency-likelihood grammar* [40], and speech recognition systems (for example [131, 62]) are other examples of using  $n$ -grams.

## 2.4 Incomplete theories and machine learning

As mentioned in the introduction to this chapter, machine learning researchers have also looked at the problem of undergeneration and there are clear links between machine learning and parsing. *Explanation-based learning* (EBL) can be viewed as being similar to parsing: in EBL, the domain theory is used to prove that an example is indeed an example. Likewise, in parsing, the grammar is used to prove a string is a sentence. Researchers in machine learning face similar problems as do researchers in NLP and call undergeneration the *incomplete theory problem* (errors of omission) and overgeneration the *inconsistent theory problem* (errors of commission) [82]. It is therefore also useful to consider related machine learning work. Rajamoney and DeJong classify and Ourston and Mooney give an overview of approaches dealing with these problems [101, 93]. Here, two sample approaches are described with a view to seeing how well they can be used to overcome undergeneration.

### 2.4.1 EITHER

Ourston and Mooney's EITHER system (Explanation-based and Inductive Theory Extension and Revision) deals with the problems of inconsistency and incompleteness [101]. EITHER uses Horn clauses as its knowledge representation and makes uses of both positive and negative examples. The incomplete theory problem is dealt with by trying to prove, using EBL, the positive example. When the proof fails, the resulting partial explanation is then examined and assumptions are tentatively removed. If no negative examples are proven after assumption removal, the assumptions are permanently removed. If negative examples are also proved, inductive methods are used to learn rules that make this discrimination between positive

and negative examples. The authors also give a similar method for dealing with inconsistent theories. Relating this work to approaches dealing with undergeneration suggests that the main difference is the use of negative examples to determine the utility of the newly learnt rules. If we recall, NLP systems generally do not encounter non-sentences that are marked as such and hence EITHER cannot be directly applied. To apply EITHER directly would be equivalent to grammar relaxation, thereby increasing overgeneration.

### 2.4.2 Fawcett

Fawcett's approach does not rest upon negative examples [32]. Like EITHER, his system generates explanations and when a concept cannot be proved, the resulting partial explanations form the basis of the search for plausible explanations. The search is limited by two constraints. Firstly, certain relations in the domain theory are considered to be necessary to be proved (unlike EITHER which can remove any assumption) and secondly, heuristics are used to arbitrate between competing explanations. These heuristics include succinctness, minimising the use of unverified assumptions, preferring specific rules over less specific rules, and so on. Heuristics are composed in an ad-hoc manner to give explanations a score. Rules are generated that allow the failing rule's antecedent to match the concept. Fawcett gives no evaluation of his system. To be useful for overcoming natural language undergeneration, Fawcett's approach will need a different set of ordering heuristics appropriate for natural language. The approach seems to assume that only a single rule is necessary for each unproved antecedent to complete the otherwise incomplete explanation. Such an approach for grammar rule learning would create flat trees, which are arguably implausible. Fawcett's system, like EITHER, assumes that examples are recognised as being positive (or negative). NLP systems do not have this information and must at times reject word salad. If Fawcett's system were used directly, the system would overgenerate by learning rules for bad sentences.

## 2.5 Discussion

In chapter one, it was argued that an approach dealing with a grammar's undergeneration should correct sentences that need correction, learn grammar to generate sentences that do not need correcting, minimise both a grammar's undergeneration and overgeneration, assign plausible parses to sentences not in  $L(G)$  and finally should produce an explicit grammar. A system meeting these criteria would have a sentence correcting and a grammar learning module. None of the systems surveyed fully met these criteria or allowed the possibility of these criteria being met.

NOMAD corrects sentences that need correcting, but is reliant upon a domain theory for success. It would be a stronger (corrective) approach to undergeneration if NOMAD had a performance theory (that is, why people make performance errors). By not differentiating between the sentence types,<sup>9</sup> NOMAD cannot be used in conjunction with a learning module and so fails to minimise future under- and overgeneration.

Weischedel corrects correctly and allows the possibility of being used with a learning module. The parses produced will not necessarily be plausible as sentence correction is achieved by correcting the parse tree in such a manner that a parse is produced, irrespective of what the parse actually says about the grammatical structure of the sentence in question. The use of meta-rules is the most interesting aspect of his work.

Parse fitting is close architecturally to the ideal system dealing with undergeneration in that the core grammar can be considered a competence grammar and the fitting procedure can be considered a correcting approach. However, the approach assumes that the core grammar does not undergenerate and treats all cases of undergeneration as errors. Likewise, the Carbonell family of approaches do not recognise this distinction between knowing when to learn and knowing when to correct and so cannot be used in conjunction with a learning module. Mellish's approach suffers from the same problem. However, his reconstruction addresses issues such as the need to carry out search, using declarative 'meta-rules', and using a unification-based formalism. As such, Mellish's method would be the choice of a

---

<sup>9</sup>That is, whether the sentence is in  $C$ ,  $P$ , or elsewhere.

system architect designing a sentence correction method.

Berwick learns grammar correctly (with respect to the criteria for successful treatment of undergeneration), but his approach is purely deductive and so does not compensate for informant incompleteness. Unlike the inductive grammar learners, he does take seriously the idea of language identification.

Vanlehn and Ball do not recognise differing sentence types and so learn grammars that will overgenerate. Because their system places strong restrictions upon the format of rules, it cannot be considered as a serious solution to learning plausible natural language grammars.

The DACS system does not recognise the boundaries between sentence types and so does not know when to correct sentences. The learning algorithm attempts to overcome induction's unsoundness by ordering the text by length, which is a crude approximation to ordering by syntactic complexity. DACS's informant will therefore be incomplete and will not guarantee that natural languages are identified in the limit. Furthermore, ordering precludes dealing with interactive language processing tasks. However, DACS does use a unification-based formalism, which sets it apart from many other grammar learners.

The Inside-Outside algorithm produces a grammar that, whilst dealing with undergeneration, will tend not to minimise overgeneration and so cannot be used easily in conjunction with a sentence correction component. Plausible parses are arguably not produced for sentences. Because the approach is not incremental, it cannot be used in an interactive application. Computationally, the algorithm is too expensive to be able to learn a grammar with a large number of non-terminals and so cannot easily be used to learn a grammar that would assign fine syntactic analyses to sentences. Again, there is no informant and so the approach cannot identify in the limit any language. However, the Inside-Outside algorithm is arguably the most attractive of the inductive grammar learners, given that it produces an explicit grammar and that it has a clear notion of convergence.

None of the incomplete theory approaches can be used directly. This is because they either use negative examples (which most NLP systems do not have access to), or are unconcerned with the plausibility of the resulting theory. As should be clear, plausibility is an important aspect of a natural language grammar.

In conclusion, most of the sentence correction approaches preclude grammar correction, and conversely, most of the grammar correction approaches preclude sentence correction. This means that the criteria for successful treatment of undergeneration are not met by any of the systems survey in this chapter. The next chapter presents an approach that meets the criteria for successful treatment of undergeneration.

## Chapter 3

# The Grammar Garden

### 3.1 Introduction

This chapter describes the learning system (whose implementation is called *The Grammar Garden*).<sup>1</sup> Conceptually, the system consists of the following components:

- A set of knowledge sources. This consists of various aspects of syntactic and lexical information that the system makes use of.
- A rule construction mechanism. The rule constructor extends the grammar by building missing rules required to complete a parse for some sentence.
- A control strategy. Deciding which rules are to be constructed is equally important as the actual construction of the rules and this is carried out by the control strategy.

Each of these components will be described in detail in section 3.2. The section following this, 3.3, gives a worked example that helps to explain how grammars are learnt. Section 3.4 outlines the implementation of the learner. Section 3.5 discusses properties of the learner and concludes the chapter.

### 3.2 System overview

Here, the various components of the learner are described.

---

<sup>1</sup>Early versions of the learner have been described in two papers [90, 89].



### 3.2.1 The set of knowledge sources

Like most *knowledge-based systems*, the grammar learner contains various sources of knowledge. In this case, they relate to the syntax of a natural language. Without these sources of knowledge, the system would not be able to learn grammar. The knowledge sources consists of:

- A grammar  $G$ .
- A lexicon.
- A language model LM.
- A model of grammaticality MG.
- A set of rule templates called *super rules*.

Each of these elements will now be described in turn.

#### The grammar $G$

The learning system is designed to overcome the undergeneration of grammar  $G$ . Although  $G$  may be empty, usually, it will contain some rules. Apart from consideration of the set of strings generated by  $G$ , another aspect of  $G$  is the *formalism* used to encode such a grammar. Grammar formalisms are languages to describe grammars. Shieber presents the following three criteria of grammar formalisms [115]:

- Linguistic felicity. How closely linguistic phenomena can be stated as linguists would want to state them.
- Expressiveness. The ability to state linguistic phenomena.
- Computational effectiveness. Whether grammars expressed in some formalism can be used to generate sentences, and any computational limitations that such a formalism might present.

A useful formalism would allow grammars to be written in a manner that is natural, allow all constructs to be described, and finally, would be computationally tractable. *Unification-based* formalisms are widely used in computational linguistics

and arguably meet the above criteria well. This is in contrast to other formalisms such as a context free grammar (CFG) which might be computationally effective, but do not capture all generalisations a linguist might want to make, as will be shortly demonstrated. The grammars learnt by the Grammar Garden are therefore unification-based.

Unification-based formalisms replace the atomic non-terminal category set of formalisms such as a CFG with a set of complex *feature-structures*. A feature structure is a partial function from features to their values. For example, a mapping from a *Person* feature might be to the value *3*. This mapping is commonly written as follows:

$$\left[ \begin{array}{l} \textit{Person} \ 3 \end{array} \right]$$

Feature structures can be recursively nested, with a feature taking a feature structure as a value:

$$\left[ \begin{array}{l} \textit{Cat} \ \left[ \begin{array}{l} \textit{Person} \ 3 \end{array} \right] \end{array} \right]$$

Furthermore, two or more features within a feature structure can share values. This is known as *reenetrancy* and is shown by a numbered square box. For example:

$$\left[ \begin{array}{l} \textit{Person} \ \boxed{1} \\ \textit{Cat} \ \left[ \begin{array}{l} \textit{Person} \ \boxed{1} \end{array} \right] \end{array} \right]$$

This feature structure states that the value of the *Person* feature must be the same in both cases.

Within the grammar formalism used in this thesis, rules are of the form  $A \rightarrow \alpha$ , where  $A$  is a feature structure and  $\alpha$  is a string of feature structures of any length. The set of features used that defines the set of possible feature structures is fixed. Fixing the set of features is an assumption made in this thesis. This, and other assumptions, are discussed in chapter seven. A typical rule might be:

$$\left[ \begin{array}{l} \textit{Number} \ \boxed{1} \\ \textit{Cat} \ \textit{NP} \end{array} \right] \rightarrow \left[ \begin{array}{l} \textit{Cat} \ \textit{Det} \end{array} \right] \left[ \begin{array}{l} \textit{Number} \ \boxed{1} \\ \textit{Cat} \ \textit{N1} \end{array} \right]$$

This is intended to say that a NP consists of a determiner followed by a nominal phrase and that the nominal category in the rule's right hand side (RHS) must agree in number with the rule's left hand side (LHS) category. To achieve agreement (for example) in a CFG requires that the agreement be stated explicitly:

$$NP1sing \rightarrow Det N1sing$$

$$NP2plu \rightarrow Det N2plu$$

Note the parsimonious nature of the single unification-based rule compared with the profligate nature of the CFG rules.

Feature structures can be ordered by how informative they are. The feature structure:

$$\left[ \begin{array}{l} Person \\ 3 \end{array} \right]$$

is more informative than the feature structure:

$$\left[ \begin{array}{l} \end{array} \right]$$

and this ordering is known as *subsumption*. Before defining subsumption more formally,<sup>2</sup> let the notation  $D(f)$  mean the value of feature  $f$  in feature structure  $D$  and let  $dom(D)$  be the domain of feature structure  $D$  (i. e. its set of features). Let a *path* in a feature structure be a sequence of features that terminates in a feature value. Given these definitions, feature structure  $D$  subsumes feature structure  $D'$  if and only if  $D(l) \subseteq D'(l)$  for all  $l \in dom(D)$  and  $D'(p) = D'(q)$  for all paths  $p$  and  $q$  such that  $D(p) = D(q)$ . Atomic feature structures only subsume identical atomic feature structures and reentrant features can subsume any feature structure.  $D$  subsumes  $D'$  is written as  $D \sqsubseteq D'$ .

The *unification* of feature structures  $D'$  and  $D''$  is the most general feature structure  $D$  such that  $D' \sqsubseteq D$  and  $D'' \sqsubseteq D$  and is written as  $D = D' \sqcup D''$ . Unification fails if the two feature structures contain inconsistent information and a failed unification produces the inconsistent category  $\perp$ .  $\perp \sqcup D = \perp$  for any category  $D$ . The empty category  $[\ ]$  unifies with any other category.

Simple examples of unification include:

---

<sup>2</sup>This definition of subsumption and unification is taken, almost directly, from Shieber [115, p.15].

$$\begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix} \sqcup \begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix} = \begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix}$$

$$\begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix} \sqcup \begin{bmatrix} \textit{Cat NP} \\ \textit{Person } \boxed{1} \end{bmatrix} = \begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix}$$

$$\begin{bmatrix} \textit{Cat NP} \\ \textit{Person 3} \end{bmatrix} \sqcup \begin{bmatrix} \textit{Cat NP} \\ \textit{Person 2} \end{bmatrix} = \perp$$

As a notational convenience, feature structures will sometimes be represented by atomic categories. These conventions will be given when necessary.

A useful (notational) extension to the formalism is *disjunction*. Braces are conventionally used to indicate disjunction in feature structures. For example,

$$\left\{ \begin{bmatrix} \textit{Person 3} \\ \textit{Cat NP} \end{bmatrix}, \begin{bmatrix} \textit{Person 3} \\ \textit{Cat N1} \end{bmatrix} \right\}$$

is the disjunction of the two categories within the braces. This notation can be extended to deal with value disjunction:

$$\begin{bmatrix} \textit{Person 3} \\ \textit{Cat } \{\textit{NP}, \textit{N1}\} \end{bmatrix}$$

This category is another way of stating the previous disjunctive category. Disjunction really is just a notational extension as it is possible to multiply-out the disjuncts into a set of conjuncts. The following list of disjunctive feature structures properties makes this clear:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \quad (\text{distribution})$$

$$A \vee A = A \quad (\text{idempotency})$$

$$\perp \vee A = A \quad (\text{bottom})$$

$$[\ ] \vee A = [\ ] \quad (\text{top})$$

$$A \sqsubseteq B \text{ if and only if } A \vee B = B \quad (\text{interdefinability of subsumption and disjunction})$$

where  $A, B$  and  $C$  are possibly disjunctive feature structures. This set of properties is taken from Pollard and Sag [96]. The disjunctive unification of some  $(A \vee B) \sqcup (C \vee D)$  is  $(A \sqcup C) \vee (A \sqcup D) \vee (B \sqcup C) \vee (B \sqcup D)$ . Disjunctive subsumption of some  $(A \vee B) \sqsubseteq (C \vee D)$  is  $(A \sqsubseteq C) \wedge (A \sqsubseteq D) \wedge (B \sqsubseteq C) \wedge (B \sqsubseteq D)$ . Disjunction is to be interpreted

strongly in the sense that  $A \vee B$  is considered true only in the case when neither  $A$  nor  $B$  can be inferred, but the disjunction  $A \vee B$  is true [72]. Ordinary unification is almost linear in time complexity [78], whilst disjunctive unification is NP-complete [63].

The formalism presented is (apart from the notational use of disjunction) a variant of the PATR-II formalism [113]. PATR-II is intended to be a tool formalism and so can be used to simulate most other unification-based formalisms.

### The lexicon

The learner does not acquire lexical information and presumes a complete lexicon. That is, all words encountered have a lexical entry. As outlined in chapter one, this is another assumption made by this research.

### The language model LM

The language model is the basis of grammaticality used by the data-driven learner, as explained in chapter five.

### The model of grammaticality MG

The model of grammaticality forms the basis of grammaticality for the model-based learner, as explained in chapter four.

### Super rules

*Super rules* are templates that correspond (roughly) to various kinds of missing rules from G. In the Grammar Garden, they consist of the following unification-based rules:

$$\begin{aligned} [ ] &\rightarrow [ ] [ ] \quad (\text{binary}) \\ [ ] &\rightarrow [ ] \quad (\text{unary}) \end{aligned}$$

The binary super rule says that any category can be written as any two categories. The unary super rule says that any category can be re-written as another category. The super rules, being completely vacuous, subsume all unary and binary rules that the grammar can express. Rules learnt are refinements of these super rules. The

super rules exploit the expressive power of a unification-based grammar, implicitly representing all unary and binary rules, and hence are complete. Note that using just unary and binary super rules is sufficient to be able to learn rules to generate any sentence.<sup>3</sup> Unary and binary super rules are not necessarily sufficient to produce rules that always assign plausible parses for sentences. Linguists typically use rules without a RHS category to introduce gaps into parse trees for constructions such as preposed constituents. For the learner to acquire such rules requires using a 0-ary super. However, this would greatly increase the search space size. Hence, gapping rules are not learnt. Likewise, linguists typically use rules with a RHS containing three categories when generating constructs such as ditransitive VPs. Again, this would imply that the learner used a trinary super rule, thereby increasing the search space. This assumption, of only learning unary and binary rules, is discussed in chapter seven.

Super rules are similar to Weischedel's meta-rules (which correspond to various kinds of syntactic error) in that they correspond to various rules missing from the grammar. However, unlike meta-rules (which explicitly enumerate the type of error that the system deals with), super rules implicitly represent all the missing (unary and binary) rules. Hence, the super rules are complete. By comparison, the set of meta-rules will most likely be incomplete. The super rules are also similar to Mellish's generalised rules. However, Mellish's rules simply join local trees together. The super rules on the other hand can be used to generate local trees. Hence, the super rules have no reliance upon the initial grammar.

### 3.2.2 The rule construction mechanism

Rule construction within the learner consists of determining what the LHS of the rule is, given a RHS. Such a determination requires a theory of how rule LHSs relate to rule RHSs. One such theory, which is popular in many linguistic theories, is *X-bar Syntax* [55]. There are many variations of X-bar Syntax. Broadly speaking, X-bar

---

<sup>3</sup>Any string of terminals can be generated using just unary and binary rules. For example, using some binary rule would rewrite a string of length  $n, n > 2$  to a string of length  $n - 1$ . Repeating this rewriting process would eventually result in a string of length 1. Strings of length 1 can be rewritten using some unary rule.

syntax constrains rules either to be of the form:

$$H[BAR\ N] \rightarrow \alpha\ H[BAR\ N]\ \beta$$

(which is recursive), or of the form:

$$H[BAR\ N+1] \rightarrow \alpha\ H[BAR\ N]\ \beta$$

(which is non-recursive). Here, the notation  $H[BAR\ N]$  indicates any category with a BAR level of  $N$ . The  $H$  category within the RHS is called the *head* of the phrase. The head characterises that phrase. For example, the head of a noun phrase is a noun. The LHS category is called the *projection* of the head. Lexical categories usually have a bar level of zero and intermediate phrases have projections that raise this bar level. Bar levels can only be raised up to some limit and a category whose bar level is equal to this limit is called the *maximal projection* of the phrase. For example, the maximal projection of a noun is a noun phrase. X-bar syntax therefore helps determine what the LHS of the rule should be and places a limit upon the number of rules that might be learnt. So, the rule  $VP \rightarrow N1$  would not be created as it has a projection that is not related to the head. X-bar syntax is therefore a powerful constraint upon rules.

X-bar syntax constitutes a restriction over the space of possible rules that are linguistically plausible. From a learning perspective, this restriction can be thought of as an *inductive bias* [50]. An inductive bias is a constraint upon the space of possible hypotheses that the learner might consider. We saw in §2.3.4 that Briscoe and Waegner used X-bar syntax as an inductive bias upon the Inside-Outside Algorithm. Likewise, and independently [88, p.40], we also use X-bar syntax as an inductive bias.

X-bar syntax relies upon locating the head in the RHS and determining if the rule is recursive, or finite. This constitutes knowledge of syntax. As such, implementations of this theory of syntax will in practice be incomplete. That is, they will not always be able to determine if a rule is finite or recursive, or what the head of the rule is. Problems of determining what the LHS should be are tackled by creating the *disjunction* of the possible categories that might form the LHS category.<sup>4</sup> Later,

---

<sup>4</sup>This is similar to the INDICO system's approach to LHS construction [121].

when sufficient evidence has built up, the LHS can be refined to be that disjunct that has proved itself to be the projection of the rule. Chapter five explains how rule LHSs are refined.

In the learner, unary rules are constructed as follows. Given some RHS of the form  $A[BAR\ X]$ , the rule that the learner will initially construct is:

$$A_i[BAR\ X+1] \rightarrow A[BAR\ X]$$

If the bar level of category  $A$  exceeds the maximal bar level, then it is not used in the rule LHS. If no category can be used in the LHS then the rule is not created. Note that unary rule construction only creates non-recursive rules. This is because recursive unary rules lead to parser nontermination.

For some RHS of the form:

$$A_1[BAR\ X]A_2[BAR\ Y]$$

the learner initially constructs the rule:

$$\{A_1[BAR\ \{X, X+1\}], A_2[BAR\ \{Y, Y+1\}]\} \rightarrow A_1[BAR\ X]A_2[BAR\ Y]$$

Again, if the bar level of category in the LHS exceeds the maximal bar level, then it is not disjoined with the other categories.

Minor categories (such as  $[Det\ +]$ ) do not have bar levels and take no part in the rule construction process. However, some rules, for example those treating possessives, arguably have a LHS which is a minor category. An example rule might be  $Det \rightarrow NP\ POSS$  (where POSS is the lexical entry for the item “s”). The extra computational expense of allowing minor categories to be part of the LHS is such that this incompleteness with respect to possessives can be overlooked. If the LHS cannot be created, then the rule is judged to be implausible and so rejected.

Note that there is no reason why the rule constructor could not use theories of phrase structure other than X-bar syntax.

### 3.2.3 The control strategy

The task of the control mechanism is to allow the rule constructor to create rules that allow at least one parse to be created for the sentence  $W$  that the system



0	Sam	1	died	2
---	-----	---	------	---

Figure 3.1: An empty chart

is currently dealing with. The sentence  $W$  guides the learning process and, in a sense, learning could be said to be incremental. Compare this with batch-orientated approaches such as the Inside-Outside algorithm, which learn all possible rules at once. By being incremental, the learning approach only learns as much as it needs to.

The control strategy needs to be exhaustive (it should allow all ambiguous attachments to be learnt), goal directed (it should consider strings of categories that have a chance of allowing the parse to be completed) and efficient. *Chart parsers* [64] meet these demands well and when adapted, can be used as the learning control strategy. Furthermore, they are very flexible, leaving the choice of processing strategy open, do not require expensive pre-processing of the grammar and use a form of book-keeping that avoids the need to re-compute previously constructed derivation sequences. These last two advantages make chart parsing especially suitable for interleaved learning and parsing. The chief disadvantage of chart parsers is that their speed is theoretically slower in the worst case than for other algorithms, notably the LR family of parsers. But in practice, the parse time between a state-of-the-art chart parser and an LR-style parser is not that great [16, p.128].

A chart parser accepts an input string of length  $n$  and builds a data-structure known as a *chart*. A chart consists of a set of *vertices* that label the input string, starting from 0 to  $n$ . For example, the sentence:

**22** *Sam died*

would have an initial chart as shown in figure 3.1.

The parser now proceeds to build structures called *edges* that span vertices. An edge can either represent the hypothesis of a phrase starting from vertex  $i$ , or can represent the fact that a phrase has been found, spanning from vertex  $i$  to vertex  $j$ , where  $i \leq j$ . The former sort of edge is called *active* and the latter edge is called

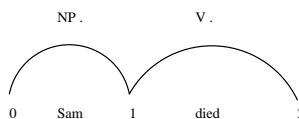


Figure 3.2: A chart with lexical edges

*inactive*. An edge is of the form:

$$\langle start, end, found, needed \rangle$$

*Start* indicates the left hand side of the edge's span, *end* indicates the right hand side of the edge's span, *found* is (roughly) a list of non-terminals that have been seen as evidence of this being a particular phrase, and *needed* is a list of categories needed before the active edge becomes inactive. An active edge has a non-empty *needed* list, whilst an inactive edge has an empty *needed* list of categories.

Initially, the parser adds *lexical edges* to the chart. A lexical edge is an inactive edge that only spans a single word. Intuitively, adding lexical edges can be thought of as 'seeding' the chart with edges that may ultimately represent phrases. Assuming the grammar:

$$S \rightarrow NP VP \quad (S)$$

$$VP \rightarrow V \quad (VP1)$$

and the lexicon:

$$Sam \mapsto NP$$

$$died \mapsto V$$

Adding lexical edges would give the chart shown in figure 3.2.

Conventionally, an inactive edge is depicted as an arc labelled with the phrasal category represented by the edge, and an active edge is depicted as a dotted pair whose left hand component is the *found* list and right hand component is the *needed* list. Active edges are normally also shown with their potential category that they might eventually 'become'. For simplicity, this extra information is omitted.

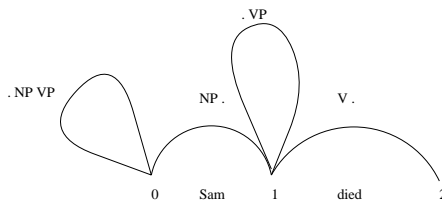


Figure 3.3: A chart after proposing edges

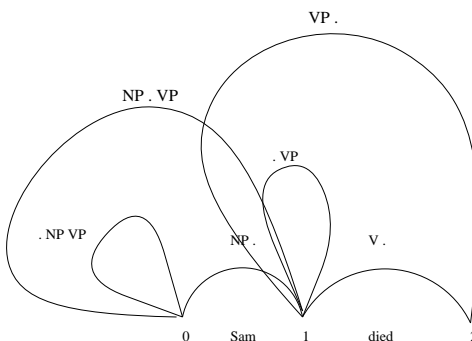


Figure 3.4: A chart after extending edges

The next step is to *propose* new active edges from inactive edges in the chart. New edges are proposed for rules whose first daughter matches with the category of an inactive edge.

Proposing edges in this way gives a bottom-up parsing algorithm. Proposing new edges to the chart so far constructed results in the chart shown in figure 3.3.

After adding edges, the final step is to *extend* active edges present in the chart. An active edge of the form  $\langle i, j, \alpha, B \beta \rangle$  can be extended if there is an inactive edge of the form  $\langle j, k, B, \text{nil} \rangle$  (for some  $i \leq j \leq k, \alpha, \beta \in N^*, B \in N$ ). Extending such an edge adds the edge  $\langle i, k, \alpha B, \beta \rangle$  to the chart. Extending the edges in the chart results in figure 3.4. An inactive edge has been added, showing that a VP has been found, along with an active edge.

For completeness, both *propose* and *extend* need to be called repeatedly until no further edges can be either added to the chart, and no edges in the chart can be further extended.

This has informally described how a chart parser works. In order to adapt the chart parser for learning, the basic chart parsing algorithm needs to be changed.

During interleaved parsing and learning, the grammar is partitioned into the grammar  $G$  and the grammar  $Q$ .  $G$  is the original grammar and  $Q$  is the learnt grammar. This is motivated on efficiency grounds (the super rules, being complete, will lead to previously learnt rules being re-learnt and hence there is no need to propose previously learnt rules). During normal parsing, both  $G$  and  $Q$  are used.

After parsing some sentence  $W$  has *failed*, the resulting chart is then ‘seeded’ with super rules at each vertex. That is, super rules are proposed. Seeding the chart allows the possibility of any unary or binary branching parse to be completed, irrespective of the original grammar  $G$ . Parsing then continues as before, except this time proposing new edges using the grammar  $G$  and the super rules. Upon instantiating the RHS of a super rule, that instantiated super rule is then passed to a *critic*

The critic is composed of a data-driven and model-driven component and is used to *refine* (or *reject*) instantiated super rules. Super rule refinement consists of using the rule constructor to create the LHS, and super rule rejection consists of discarding rules that contain categories that cannot co-exist as part of a rule’s RHS. Chapters four and five describe in detail how RHSs are refined and rejected. If the LHS cannot be constructed, or no categories can co-exist within the RHS of the rule, then the edge containing the instantiated super rule is marked as being *bad*. Edges marked as being bad are not proposed from, and are not used in parse tree. By marking edges in this way, the learner does not pursue possible parses that arise from linguistically implausible rules. If the edge is not marked as being bad, then that edge is treated as other edges, and the newly constructed rule is retained.

Parsing continues until no edges can be extended. Any rules used in a parse for  $W$  that are not subsumed by any other rule in the grammars  $G$  or  $Q$  (ignoring the super rules) are kept for later use.

After interleaved parsing and learning has concluded, the system can either be used to deal with another sentence, can be used to post-process the learnt rules in an attempt to reduce overgeneration, or can be used to merge the learnt rules with the original grammar, thereby delivering a grammar which can be used in another NLP application.

In conclusion, the control strategy is based upon a chart parser, and makes use

of a critic to determine if some string of categories should be passed to the rule constructor. The critic, as well as refining edges, helps prevent the learner from considering unfruitful branches of the search space.

### 3.3 A worked example

Here, an example will be worked through. Note that although the grammar is unification-based, atomic symbols are used instead as a notational convenience.<sup>5</sup>

Suppose the system started with the following grammar:

$$S \rightarrow NP VP \quad (S1)$$

$$NP \rightarrow Det N1 \quad (NP1)$$

$$VP \rightarrow V0 NP \quad (VP1)$$

and with the lexicon:

the  $\mapsto$  Det

cat  $\mapsto$  N1

happy  $\mapsto$  Adj

chases  $\mapsto$  V0

Sam  $\mapsto$  NP

Parsing the sentence:

**23** *Sam chases the happy cat*

will fail, producing the chart as shown in figure 3.5.<sup>6</sup> Enabling learning and adding new edges (‘seeding’) using the binary super rule,<sup>7</sup> produces the chart as shown in 3.6. That is, each inactive edge has been used to propose a new active edge that can join with an inactive edge in the next vertex. Clearly, parsing can now be resumed using these new active edges. Concentrating just on vertex 0, extending these edges produces the chart shown in figure 3.7. The extended edge (marked with a question

---

<sup>5</sup>This grammar is meant to be demonstrational only and not intended to make any linguistic claims.

<sup>6</sup>In this, and in the other diagrams of various charts, a few edges are missing. These are unimportant for the exposition.

<sup>7</sup>For simplicity, only the binary rule will be used in this example. It should be clear how unary rules are learnt.

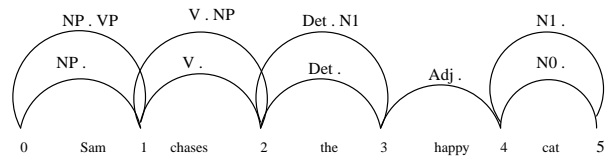


Figure 3.5: The resulting chart after a failure to parse

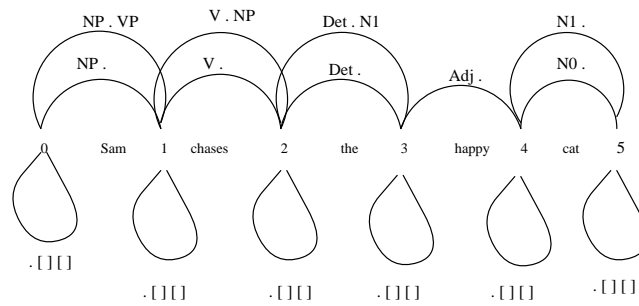


Figure 3.6: The chart after seeding with binary super rules

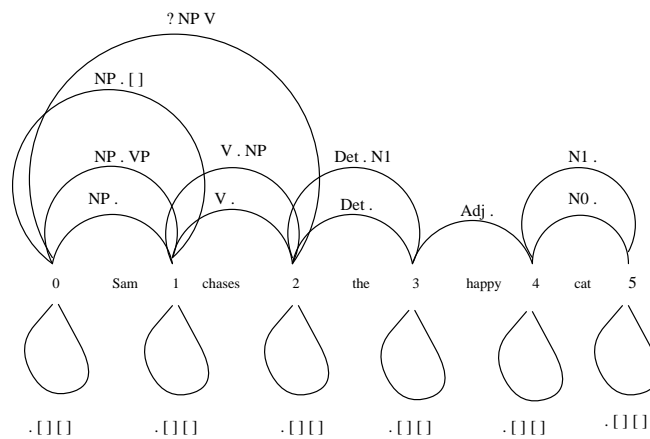


Figure 3.7: The chart after extending a super rule instantiation

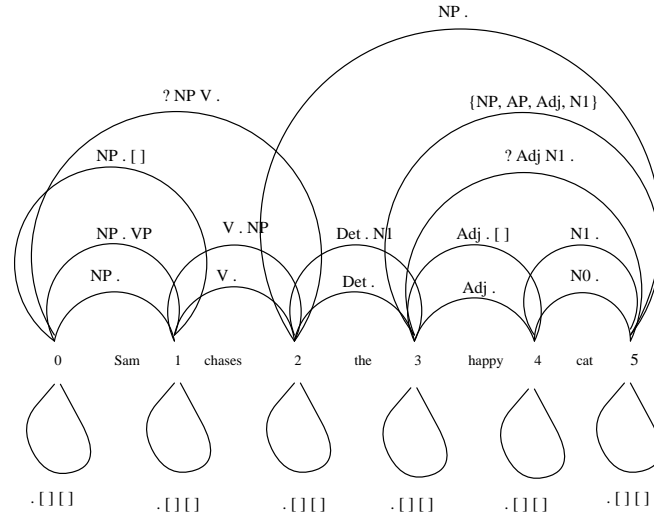


Figure 3.8: The chart after extending another super rule instantiation

mark) represents the possible application of a rule whose RHS is  $NP V$ . The task now is to decide if this RHS can be used as part of a plausible rule. Let us assume that criticising the edge containing these categories results in the edge being judged as being implausible. This will both not lead to a new rule being constructed, and will also not lead to further edges being proposed from this edge.

Suppose now the parser extended the active edge starting at vertex 3, and then tried to extend this active edge  $Adj \cdot []$  starting at vertex 3 and ending at vertex 4, with the inactive edge labelled  $N1$  spanning from vertex 4 to 5. This results in an inactive edge (marked with a question mark) being added to the chart (as shown in figure 3.8). Because the edge originated from a super rule, it is passed to the critic. Assuming that the critic judges this edge to be plausible, this edge is then passed to the rule constructor. The constructor then builds the rule:

$$\{AP, NP, Adj, N1\} \rightarrow Adj N1$$

Now, the active edge  $Det \cdot N1$  can be extended with this criticised edge, resulting in the chart shown in figure 3.8. Now, interleaved parsing and learning can continue, until finally, the parse as shown in figure 3.9 is produced. That is, the active edge spanning from vertex 1 to vertex 2 has been extended with the inactive  $NP$  edge, giving an inactive  $VP$  edge spanning from vertex 1 to vertex 5. Now, the active edge

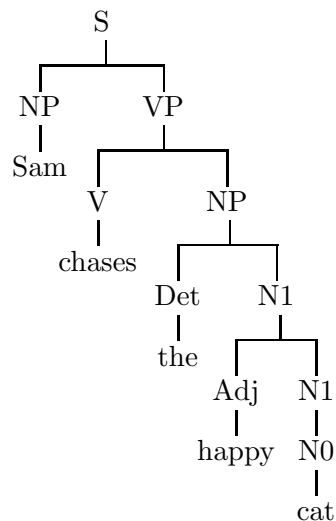


Figure 3.9: The final parse tree

spanning from vertex 0 to 1 can be extended with the VP edge, giving an inactive edge that spans the entire chart. Note that the disjunctive LHS of the learnt rule has, through unification, become a single, non-disjunctive category in the parse tree. That is,  $\{AP, NP, Adj, N1\} \sqcup N1 = N1$ , where  $N1$  is the first category within the RHS of the rule NP1. In general, parse trees will contain disjunctive categories.

### 3.4 Implementation

The system consists of 3300 lines of Common Lisp (AKCL) and has been run on a Sun 3/50, a Silicon Graphics Indigo, a Sparc Workstation and an IBM RS6000. It is embedded in the Grammar Development Environment (GDE), version 1.33 [45]. Although the GDE is a mature and useful tool for manipulating unification-based grammars, its chart parser had to be re-written for the following reasons:

- It uses non-disjunctive unification, whereas the Grammar Garden uses disjunctive unification.
- The GDE's parser does not carry scoring information necessary for the data-driven learner (see chapter five for an explanation).



- The GDE’s parser uses a monolithic grammar, but the learner uses a partitioned grammar of original rules and learnt rules.
- It only calls *propose* once for each inactive edge (therefore introducing incompleteness with respect to a dynamic grammar).

The learning system has a number of flags that allow the system to be parameterised. Here is a typical setting which shows that the system will learn, using a critic composed of a number of components (type checking, LP rules and a Head Feature convention). As it turns out, these components make-up the model-based aspect of learning, which is explained in the next chapter. Parameterising the system allows experimentation between various learning styles and produces distinct learning configurations.

```
3 Gde> flags
```

```
Current flag settings:
```

```
Learning           : ON
Type checking      : ON
LP rules           : ON
HFC                : ON
SBL                : OFF
Training           : OFF
```

Here, the implementation is shown learning the same rule as was learnt in the worked example, using the same grammar and lexicon as before.

```
9 Parse+>> Sam chases the happy cat
1 rule(s) acquired.
1 parse(s)
10 Parse+>> !*parses*
((('S1'
  ((|Sam|)
    ('VP'
      ((|chases|)
```

```

('NP1' ((|the|
          ('*binary1583' ((|happy|
                           (|cat|))))))))
11 Parse+>>

```

The new rule is used in the parse and prefixed by a star. This is the same rule as learnt in §3.3 (namely  $\{AP, NP, Adj, N1\} \rightarrow Adj\ N1$ ).

Note that the chart parser is bottom-up. Because the chart parser drives learning, learning also takes places bottom-up. With some work, learning could take place using a parser that adopted a different strategy.

### 3.5 Conclusion

This chapter presented a grammar learning system that can be used to deal with a grammar's undergeneration. Of the systems presented in chapter two, the learner is closest to Briscoe and Waegner's batch-orientated approach (see §2.3.4). They use a unification-based formalism, a data-driven component, and also a model-based component. Our incremental learner differs in that it uses a disjunctive unification-based formalism, makes greater use of the model-based component, and attempts, unlike Briscoe and Waegner, to deal with overgeneration.

The rest of this section discusses properties of the grammar learner. These properties include:

- Meeting the success criteria.
- Completeness.
- Termination.
- Complexity.

The first property is concerned with how well the learner deals with undergeneration. The other properties are concerned with computational aspects of the grammar learner.

### 3.5.1 Meeting the success criteria

If we recall from chapter one, the criteria for success consists of attempting to reduce a grammar's undergeneration, trying not to overgenerate, learning a grammar that assigns plausible parses for sentences, and finally using a grammar in a form suitable for NLP.

Undergeneration is clearly reduced by learning (even if just for the sentence that the extended grammar can now generate, but the unextended grammar could not generate).

Overgeneration is controlled through the rejection of implausible rules by the critic. For example, if when learning rules for the example sentence 23, the critic allowed rules such as  $NP \rightarrow Det\ NP$  and  $NP \rightarrow Adj\ N$ , to be acquired, then these rules would lead to overgeneration; the ungrammatical string 24, for example, would be generated by the extended grammar:

**24** \**Sam chases happy cat*

Plausibility is closely related to overgeneration. For example, the previous rules do not assign a plausible parse to sentence 23. Hence, rejecting such rules using the critic helps produce plausible parses.

By using a unification-based grammar, the learner easily meets the final criteria of using a grammar in a form suitable for NLP.

In conclusion, the grammar learner (in theory) is a solution to the problem of undergeneration. Chapter six quantitatively evaluates just how well an implementation of the theory meets the success criteria.

### 3.5.2 Completeness

The super rules express all unary and binary rules. As was shown, these are sufficient to allow rules to be learnt that will generate any sentence. Furthermore, if the critic is always able to identify when a rule is plausible, or otherwise, then that critic can be seen as a model-based informant [42]. Given such an informant, natural languages can be identified in the limit. Hence, if the model is complete, the learner is also complete with respect to learnability. In practice, the model is usually incomplete

and so the learner will be incomplete. This incompleteness can be reduced by the contribution of the data-driven component.

### 3.5.3 Termination

The learner will terminate when learning grammar for any sentence. This is because the learner does not acquire rules that allow cyclic derivation sequences to be constructed. For example, the learner does not create unary rules that are recursive. Allowing cyclic derivations would lead to parse trees that are infinitely deep.

### 3.5.4 Complexity

Grammar learning takes at least exponential space with respect to sentence length. This is because grammar learning implies that the parser, in the worst case, finds all binary and unary parses for any given sentence. As is well known, parsing with ambiguous grammars is of exponential space complexity. For example, compound nouns are usually generated using a rule such as  $N \rightarrow N N$ . This generates, with respect to sentence length, a number of parses equal to the Catalan series [26]. To overcome this intractability implies that the control strategy is made incomplete. Interesting ways of introducing incompleteness would be to introduce what Fodor, Bever and Garrett call *performance strategies* [34]. Berwick comments that [5, p.166]:

Thus, if ordinary processing methods are tailored to the most difficult situation, we can easily miss the forest for the trees -preparing for the worst cases, but missing streamlined strategies that would work in the usual case.

Performance strategies are also similar to the idea of *heuristic search* used in knowledge-based systems. In the context of the control strategy, incompleteness might be introduced by halting the learning process after constructing  $n$  parses, or halting after creating  $m$  edges. Both of these resource bounds are used in chapter six.

The next chapter explains what is meant by model-based learning in this thesis and shows how the critic can be used to learn plausible grammars. The following

chapter explains what is meant by data-driven learning and shows how data-driven learning interacts with model-based learning.

## Chapter 4

# Model-Based Learning

### 4.1 Introduction

The purpose of this chapter is to explain what a model of grammaticality is, to show how a grammar can be extended with such a model, and finally, to discuss characteristics of the model-based learner.

### 4.2 Model-based learning in machine learning and in linguistics

Recently, machine learning researchers have looked at *model-based* methods of learning. Instead of relying upon many training examples to constrain the search for some generalisation, model-based methods constrain the search by using knowledge of the task domain. This knowledge is called a *model*. After using the model to analyse an example, these methods produce a valid generalisation of the example, along with a deductive justification of the generalisation in terms of the model [82]. Because of this, model-based methods are sound in the sense that the generalisation acquired deductively follows from the model. In the context of grammar learning, the training examples will be sentences, the model will be a high-level theory of syntax, and the generalisation will be a grammar generating those sentences.

It could be argued that the model could be compiled-out to produce any generalisation required without recourse to training data. However, such a compilation process is usually computationally far too expensive to perform without guidance

from training examples. For example, the grammar presented in chapter six can express  $621495189504^3$  binary rules, and  $621495189504^2$  unary rules.<sup>1</sup> Clearly, all of these rules cannot be compiled-out and tested against the model to determine if they are plausible. Furthermore, model incompleteness will mean that implausible rules will also be compiled-out. Therefore, it is valid to say that model-based learning involves learning, but the learning of search heuristics over the concept space defined by the model [68], and not of learning concepts themselves. We shall consider what the linguistic equivalent of the model is next.

Linguists interested in human language acquisition have proposed that successful language learning is possible if the human language learner has an innate knowledge of language [21, p.51]. This innate knowledge is usually called *Universal Grammar* (UG) [22, p.29]. UG is motivated on the grounds that the training data available to children underdetermines the final grammar (the training data is too impoverished to explain why a full competence grammar is acquired), the training data may contain performance errors ('noise') and finally the data does not contain negative evidence (errors are not corrected) [128]. The last motivation is especially interesting, given the popularity of using negative examples in machine learning algorithms and the fact that, with negative examples, almost all languages are learnable in the limit. An example of this can be seen with the machine learning algorithm ID3 [98], which learns a classifier after being trained with positive and negative examples. The author trained ID3 using 60 grammatical sentences, and 100 ungrammatical sentences. After training, it was tested on 60 unseen sentences, and 100 unseen ungrammatical strings. The results showed that with negative examples, the classifier could label all the ungrammatical test strings as being ungrammatical, and most of the test sentences as being grammatical. When trained without the negative (ungrammatical) examples, the classifier considered all of the test sentences and strings to be grammatically well-formed. Negative examples are therefore a

---

<sup>1</sup>A unification-based grammar with  $n$  features, where each feature  $f_i$  has  $|v_i|$  values, can express:

$$\left(\prod_{j=1}^n \nu_j\right) \tag{4.1}$$

distinct categories. This assumes that features do not contain categories as values (in which case the space would be infinite).

powerful source of grammatical information. However, as was stated, children are not told when they use ungrammatical sentences and so this method of identifying languages in the limit is not used by language acquisition theorists. Instead, UG is usually formulated as a *model of grammaticality*, consisting of a set of *principles* and *parameters*. Principles capture generalisations across particular constructions and parameters set in the principles to operate in various ways. An example principle of UG (called *subjacency* in Government and Binding Theory (GB) [23]) might say that a filler can only be ‘moved’ from a trace across a limited number of constituent boundaries. The parameter for such a principle would be the set of constituent boundaries. In English, this set includes S and NP. For example, the following ungrammatical sentences can be explained as violations of this parameter setting for English [128, p.23]:

**25** \**What did Mary wonder whether John bought e*

**26** \**What did Mary believe the claim that John saw e*

In the first case, the filler *what* has moved across two S categories from *e* and in the second case, the filler *what* has moved across two S categories and an NP category. Language acquisition using UG consists of ‘triggering’ a particular setting of these parameters. Parameters are triggered by the learner being exposed to a training set of sentences.

Several linguistic theories are influenced by UG considerations. The most obvious example is Government and Binding Theory (GB) [23]. Other approaches, which are less obviously motivated by acquisition theory, include Lexical Functional Grammar (LFG) [7], Generalised Phrase Structure Grammar (GPSG) [38] and Head Driven Phrase Structure Grammar (HPSG) [96].<sup>2</sup> An implementation of a model of grammaticality would therefore draw upon these linguistic theories.

The model-based component of the critic (of the Grammar Garden) draws upon GPSG’s realisation of UG. Most of the parameters within the model are already set in advance (to expect English). The parameters set during learning are the

---

<sup>2</sup>Fodor has shown how GPSG can be seen in terms of language acquisition from a principles and parameters perspective [35].



individual phrase structure rules (which Fodor considers to be the parameters of GPSG). Therefore, because the model happens to draw largely upon GPSG, it would be true to say that the grammar ‘learnt’ is a GPSG grammar. However, the system is not limited just to learning GPSG-style grammars. With some work, the model could use principles drawn from other linguistic theories. For example, the system could use subadjacency. This would then mean that a GPSG grammar is not being learnt. Model-based learning is capable of learning any style of grammar, so long as the relevant principles can be formalised appropriately.

The rest of this chapter is as follows. Section 4.3 presents a unification-based grammar  $G$ . Section 4.4 describes the components of the model of grammaticality. Section 4.5 then gives various examples of grammar  $G$  being extended using model-based learning. Finally, section 4.6 is a concluding discussion.

### 4.3 The initial grammar and lexicon

The following grammar is adapted from the example grammar “ex-sem” given in the third release of the Alvey Tools Grammar and is intended to be for demonstrational purposes. That is, the exact features structures used are unimportant. Note that the grammar contains a rule (PP) that cannot be used in any derivation sequence constructed using  $G$ . This rule is intentionally in  $G$  and is used in one of the learning examples.

The features of the categories are:

Feature	Values
N	+ -
V	+ -
BAR	0 1 2 3
DET	+ -
PER	1 2 3
PLU	+ -
PRD	+ -
NTYPE	NAME PRO COUNT MASS
DEF	+ -
SUBCAT	INTRANS TRANS DITRANS
PAST	+ -
CASE	NOM ACC
VFORM	FIN PASS BSE
ADV	+ -
PFORM	TO BY
AUX	DO BE -
INV	+ -
NULL	+ -
EMPTY	+ -
CONJ	AND BOTH BUT NEITHER NOR OR NULL

The rules are:

$$\begin{bmatrix} N - \\ V + \\ \text{BAR } 2 \\ \text{DET } - \\ \text{PER } \boxed{1} \\ \text{PLU } \boxed{2} \\ \text{PAST } \boxed{3} \\ \text{VFORM } \text{FIN} \\ \text{AUX } \boxed{4} \\ \text{INV } \boxed{5} \end{bmatrix} \rightarrow \begin{bmatrix} N + \\ V - \\ \text{BAR } 2 \\ \text{DET } - \\ \text{PLU } \boxed{2} \\ \text{PRD } - \\ \text{CASE } \text{Nom} \\ \text{PER } \boxed{1} \end{bmatrix} \begin{bmatrix} N - \\ V + \\ \text{BAR } 1 \\ \text{DET } - \\ \text{PER } \boxed{1} \\ \text{PLU } \boxed{2} \\ \text{PAST } \boxed{3} \\ \text{VFORM } \text{FIN} \\ \text{AUX } \boxed{4} \\ \text{INV } \boxed{5} \end{bmatrix} \quad (\text{S1})$$

(paraphrase:  $S \rightarrow NP VP$ )

$$\begin{array}{c} \left[ \begin{array}{l} N - \\ V + \\ BAR\ 1 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ PAST\ \boxed{3} \\ VFORM\ \boxed{6} \\ AUX\ \boxed{4} \\ INV\ \boxed{5} \end{array} \right] \end{array} \rightarrow \begin{array}{c} \left[ \begin{array}{l} N - \\ V + \\ BAR\ 0 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ SUBCAT\ Intrans \\ PAST\ \boxed{3} \\ VFORM\ \boxed{6} \\ AUX\ \boxed{4} \\ INV\ \boxed{5} \\ Conj\ Null \end{array} \right] \end{array} \quad (VP1)$$

(paraphrase:  $VP \rightarrow V0$ )

$$\begin{array}{c} \left[ \begin{array}{l} N - \\ V + \\ BAR\ 1 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ PAST\ \boxed{3} \\ VFORM\ \boxed{6} \\ AUX\ \boxed{4} \\ INV\ \boxed{5} \end{array} \right] \end{array} \rightarrow \begin{array}{c} \left[ \begin{array}{l} N - \\ V + \\ BAR\ 0 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ SUBCAT\ Trans \\ PAST\ \boxed{3} \\ VFORM\ \boxed{6} \\ AUX\ \boxed{4} \\ INV\ \boxed{5} \\ Conj\ Null \end{array} \right] \end{array} \quad \begin{array}{c} \left[ \begin{array}{l} N + \\ V - \\ BAR\ 2 \\ DET - \\ PRD - \\ CASE\ Acc \end{array} \right] \end{array} \quad (VP2)$$

(paraphrase:  $VP \rightarrow V0\ NP$ )

$$\begin{array}{c} \left[ \begin{array}{l} N + \\ V - \\ BAR\ 2 \\ DET - \\ PER\ \boxed{1} \end{array} \right] \end{array} \rightarrow \begin{array}{c} \left[ \begin{array}{l} DET + \\ Conj\ Null \end{array} \right] \end{array} \quad \begin{array}{c} \left[ \begin{array}{l} N + \\ V - \\ BAR\ 1 \\ DET - \\ PER\ \boxed{1} \end{array} \right] \end{array} \quad (NP1)$$

(paraphrase:  $NP \rightarrow Det\ N1$ )

$$\begin{bmatrix} N + \\ V - \\ BAR\ 1 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ PRD\ \boxed{3} \\ Ntype\ \boxed{4} \\ CASE\ \boxed{5} \end{bmatrix} \rightarrow \begin{bmatrix} N + \\ V - \\ BAR\ 0 \\ DET - \\ PER\ \boxed{1} \\ PLU\ \boxed{2} \\ PRD\ \boxed{3} \\ Ntype\ \boxed{4} \\ CASE\ \boxed{5} \\ Conj\ Null \end{bmatrix} \quad (N1)$$

(paraphrase:  $N1 \rightarrow N0$ )

$$\begin{bmatrix} N - \\ V - \\ BAR\ 2 \\ DET - \end{bmatrix} \rightarrow \begin{bmatrix} N - \\ V - \\ BAR\ 0 \\ DET - \\ Conj\ Null \end{bmatrix} \quad \begin{bmatrix} N + \\ V - \\ BAR\ 2 \\ DET - \end{bmatrix} \quad (PP)$$

(paraphrase:  $PP \rightarrow P\ NP$ )

The lexical entries are:

$$\begin{aligned}
cat &\mapsto [N +, V -, BAR\ 1, DET -, PER\ 3, PLU -, NTYPE\ COUNT] \\
chases &\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, DET -, PER\ 3, PLU -, SUBCAT\ TRANS \\ PAST -, VFORM\ FIN, AUX -, INV -, CONJ\ NULL \end{array} \right] \\
down &\mapsto [N -, V -, BAR\ 0, DET -, SUBCAT\ NP, CONJ\ NULL] \\
happy &\mapsto [N +, V +, BAR\ 1, DET -, ADV -] \\
road &\mapsto [N +, V -, BAR\ 0, DET -, PER\ 3, PLU -, NTYPE\ COUNT, CONJ\ NULL] \\
Sam &\mapsto [N +, V -, BAR\ 2, DET -, PER\ 3, PLU -, PRD -, NTYPE\ NAME] \\
the &\mapsto [DET +, DEF +, CONJ\ NULL]
\end{aligned}$$

Note that the entries with BAR levels of one arguably should have a BAR level of zero. However, having a BAR level of one simplifies the exposition. One could interpret BAR one categories in the lexicon as representing a BAR level one category re-written implicitly by a rule that raises the BAR level by one. Note also, the entry for the word *happy* cannot be generated by the grammar  $G$ . Again, this is intentional and rules will be learnt that enable  $G$  to generate sentences that contain this word.

## 4.4 The model

The model consists of the set of principles  $P_1 P_2 \dots P_n$ . Each principle  $P_i$  is a predicate over a rule  $\alpha$  implicitly used in an edge and is true if  $P(\alpha)$  cannot be proved to be false:

$$\bigwedge_{i=1}^n P_i(\alpha) \quad (4.2)$$

If some  $P_j$  cannot prove  $\alpha$  to be implausible,  $P_j(\alpha)$  is judged as being plausible. This is motivated on the grounds that, in practise, the principles will be incomplete with respect to being able to prove if  $\alpha$  is linguistically plausible, or otherwise. However, the data-driven component of the critic, which is explained in the next chapter, is complete, and hence can deal with some  $\alpha$  passed-on by the model-based component of the critic. If the principles rejected  $\alpha$  because they failed to prove it to be linguistically plausible, the data-driven component of the learner would not have a chance to compensate for incompleteness in the model of grammaticality. The actual choice of the principle set depends upon decisions such as capturing generalisations that are thought to hold across languages. In this thesis, the choice of principles is ad hoc, and not supposed to be a statement about being the ‘best’ such set. The principle set in this thesis consists of:

- Linear precedence rules.
- Types.
- Feature-passing conventions

These are all drawn from GPSG. Other principles, from other linguistic theories, could be used. Indeed, an extension of this work, as discussed in the final chapter, would be to extend the principle set. The system is so designed that the principle set used in learning can be any combination of these principles (allowing flexibility in experimentation). There are other, inbuilt principles used in the learner, such as X-bar syntax. However, these cannot be varied in the same manner. For example, learning without X-Bar syntax would result in very low quality rules being constructed. Interestingly enough, Gazdar *et al* argue for inbuilt principles, and not for variable principles [38, p.3]. From an experimental perspective however, it is

preferable to have principles that can be varied. Like axioms, they can be changed, and the effects then noted.

The principles that can be varied will now be explained in turn.

#### 4.4.1 Linear precedence rules

A PSG may contain rules such  $A \rightarrow B C$  and  $A \rightarrow B D$  which are used to generate trees such as  $(A (B C))$  and  $(A (B D))$ . Within the tree  $(A (B C))$ , the daughter B is said to *linearly precede* daughter C. Likewise in tree  $(A (B D))$ , daughter B linearly precedes daughter D. A linear precedence (LP) constraint between two daughters A and B is described by the notation  $A \prec B$ . This means that B cannot linearly precede A in a tree and hence a rule cannot be constructed that would license such a tree. Given such a LP rule, rules such as  $A \rightarrow B A$  would be ill-formed. LP rules have a history in linguistics (described by Gazdar *et al* [38, p.47]) and are used to express intra-language generalisations. For example, within GPSG, lexical categories subcategorise, whilst phrasal categories do not subcategorise and, according to GPSG, lexical categories linearly precede phrasal categories in rules. The LP rule:

$$[\text{SUBCAT}] \prec \sim [\text{SUBCAT}]$$

expresses this fact. For this rule to be violated, a daughter must both not subcategorise and also linearly precede a subcategorising daughter. For example, such an LP rule would be satisfied by a local tree  $(\text{NP} (\text{DET the}) (\text{N1 cat}))$ , but not by the local tree  $(\text{NP} (\text{N1 cat}) (\text{DET the}))$ .

The model contains the following LP rules:

Rule name	rule
LP1	$[\text{SUBCAT}] \prec \sim [\text{SUBCAT}]$
LP2	$\begin{bmatrix} N + \end{bmatrix} \prec \begin{bmatrix} N -, V -, BAR 2 \end{bmatrix}$
LP3	$\begin{bmatrix} N + \end{bmatrix} \prec \begin{bmatrix} N -, V +, BAR 2 \end{bmatrix}$
LP4	$\begin{bmatrix} N -, V -, BAR 2 \end{bmatrix} \prec \begin{bmatrix} N -, V +, BAR 2 \end{bmatrix}$

LP1 has been previously described. LP2 says that prepositional phrases follow nominal, adjectival, or adverbial phrases. For example, the sentence:

**27** *\*the in the park boy laughed*

is ungrammatical as a prepositional phrase cannot precede a noun. LP3 says that nominal, adjectival, or adverbial phrases precede VPs or sentences. This accounts for the ungrammaticality of sentences such as :

**28** *\* laughed in the park the boy*

LP4 says that PPs precede VPs or sentences, but not the other way around.

#### 4.4.2 Types

As well as restricting the linear ordering of categories in trees (and hence restricting the ordering of categories in rules), it is also useful to restrict the co-occurrence of categories within rules. For example, one may want to state that determiners can co-occur in the RHS of a rule with a nominal, adverbial, or adjectival category, but not with a NP:

**29** *Sam chases the cat*

**30** *\*Sam chases the Sam*

Although this is possible to achieve using LP rules:

$$\begin{aligned} \left[ \begin{array}{c} DET + \\ \end{array} \right] &\prec \left[ \begin{array}{c} N +, V -, BAR 1 \\ \end{array} \right] \\ \left[ \begin{array}{c} DET + \\ \end{array} \right] &\prec \left[ \begin{array}{c} N +, V -, BAR 2 \\ \end{array} \right] \\ \left[ \begin{array}{c} N +, V -, BAR 1 \\ \end{array} \right] &\prec \left[ \begin{array}{c} DET + \\ \end{array} \right] \end{aligned}$$

(that is, the first of these LP rules allows determiners to combine with a nominal category, the second LP rule says that determiners cannot be preceded by a NP, whilst the third LP rule says that NPs cannot be preceded by a determiner.) it is also possible to state this co-occurrence more compactly by associating *types*<sup>3</sup> with each syntactic category and determining if these types can be (functionally) applied together. This is similar to type checking in programming languages. To perform this checking requires a language to express these types [29, p.88]. Such a language is the extensionally typed lambda calculus [29, p.89], which is defined as follows:

---

<sup>3</sup>Types in computational linguistics have a primary use in semantic interpretation. Enforcing co-occurrence restrictions using types is the way we use them.

- $e$  is a type.
- $t$  is a type.
- If  $a$  and  $b$  are types, then  $\langle a, b \rangle$  is a type.
- Nothing else is a type.

Drawing upon GPSG, this typed language “represents the semantic role of the various syntactic categories in the grammar” [38, p.185]. Hence, a failure to apply the types corresponding to some pair of syntactic categories implies that these syntactic categories cannot co-occur in the same rule. For example, if the type of a determiner is:

$$\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$$

the type of a nominal category is:

$$\langle e, t \rangle$$

and the type of a NP is:

$$\langle \langle e, t \rangle, t \rangle$$

then the type of determiners can be applied with the type of nominal categories:

$$\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle \circ \langle e, t \rangle = \langle \langle e, t \rangle, t \rangle$$

(where  $\circ$  represents the functional application operator). However, the type of determiners cannot be applied to the type of NPs. Hence, given just these types, it is possible to restrict the co-occurrence of determiners. This is achieved more compactly than when using the previous LP rules. Note that types complement, and do not replace, the LP rules in that linear precedence cannot be enforced with type checking.

Type checking is implemented as follows. Let  $TYP$  be a partial function from feature structures to types. If  $A$  and  $B$  are feature structures within the RHS of a rule, then if

$$TYP(A) \circ TYP(B)$$



is defined, then categories  $A$  and  $B$  can co-occur within the RHS of a rule.

The set of category-type pairs extensionally defining the  $TYP$  function in the model are:

category	type
$[V +, N -, BAR\ 2, DET -]$	$t$
$[V +, N -, BAR\ 1, DET -]$	$\langle\langle e, t \rangle, t \rangle, t\rangle$
$[V +, N -, BAR\ 0, DET -, SUBCAT\ INTRANS]$	$\langle\langle\langle e, t \rangle, t \rangle, t \rangle$
$[V +, N -, BAR\ 0, DET -, SUBCAT\ TRANS]$	$\langle\langle\langle e, t \rangle, t \rangle, \langle\langle\langle e, t \rangle, t \rangle, t \rangle\rangle$
$[N +, V -, BAR\ 2, DET -, PRD -]$	$\langle\langle e, t \rangle, t \rangle$
$[N +, V -, BAR\ 1, DET -]$	$\langle e, t \rangle$
$[N +, V -, BAR\ 0, DET -]$	$\langle e, t \rangle$
$[N +, V +, DET -]$	$\langle\langle e, t \rangle, \langle e, t \rangle\rangle$
$[DET +]$	$\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle\rangle$

These types are for demonstrational purposes only and not meant to be taken as a formal claim of the co-occurrence patterns of categories in rules for English syntax.

In practice, within the system, not every syntactic category will have a corresponding type (due to incompleteness) and hence, if  $TYP(A)$  is undefined, then  $TYP(A) \circ TYP(B)$  is taken as being defined. It would then be the task of other aspects of the model to reject a super rule instantiation that satisfied type checking through incompleteness.

#### 4.4.3 Feature-passing conventions

Many of the features of a grammar are *head features*, whose distribution involves the head of a rule. For example, a local tree generated using a rule with a nominal head would have a mother containing the same PLU feature as the nominal head daughter. This is expressed using a *head feature convention* (HFC) [38, p.94], which places a restriction upon local trees as follows:

$$\phi(C_0) \mid Head = \phi(C_H) \mid Head \quad (4.3)$$

$C_0$  is the LHS category of some rule,  $C_H$  is the head category of that rule,  $\phi$  is a mapping from categories in rules to nodes in trees,  $\mid$  is function restriction and  $Head$  is the set of head features. Note that this is the simplest possible head feature

convention. There are many other, more realistic conventions possible (for example those involving multiple heads). However, this HFC is sufficient for demonstrational purposes.

Within the model of grammaticality, the head features are all the previously mentioned features (in §4.3) except for the set of features

$$\{\text{NTYPE, CASE, CONJ, NULL, BAR}\}$$

Features not in the set of head features do not obey the HFC. In particular, as outlined in §3.2.2, the BAR feature receives a special treatment.

The HFC is implemented as follows. Instead of being a distinct device from the grammar (as in GPSG and HPSG) and applying the HFC to local trees, the HFC is compiled directly into rules. Those features that are in the head set are shared between the LHS and the head category, whilst those features that are not in the head set (with the exception of the BAR feature) are removed from the LHS. Compiling the HFC into rules is an efficiency measure and is used by other systems, such as the Grammar Development Environment (GDE) [18]. A rule obeys the HFC if the head features are shared between the head and the projection and the non-head features are not shared between the head and the projection. For example, the rule:

$$\begin{bmatrix} N + \\ V - \\ BAR\ 2 \end{bmatrix} \rightarrow \begin{bmatrix} N - \\ V + \\ BAR\ 1 \end{bmatrix}$$

violates the HFC as the N feature is not shared between the head and the projection.

This concludes the description of the model of grammaticality. The next section shows how this model can be used to learn plausible rules.

## 4.5 Model-based rule learning

This section demonstrates the capabilities of model-based learning by firstly representing the learning example given in chapter three, showing the effect of gradually increasing the extent of the model's coverage, secondly by giving an example of ambiguous attachment, thirdly by showing how ungrammatical sentences can be

detected, and finally by presenting an example where both unary and binary rules are acquired.

Suppose the Grammar Garden's parser tried to parse the sentence:

**31** *Sam chases the happy cat*

using just the initial grammar and with the learning facility turned off:

```
: Entering Grammar Garden Parser ... (level 2)
6 Parse+>> Sam chases the happy cat
0 parse(s)
850 msec CPU, 1000 msec elapsed
```

As the rule  $N1 \rightarrow AdjP\ N1$  is missing from the initial grammar, the sentence cannot be parsed. We therefore need to deal with this.

Now, if the sentence is reparsed using just the binary super rule, without model-based learning, then we have the following result:

```
21 Parse+>> Sam chases the happy cat
learning
15 rule(s) acquired.
244 parse(s)
91863690 msec CPU, 98998000 msec elapsed
```

No instantiations of the super rule are rejected by the model (since the model of the critic has been turned-off), and the system acquires 15 rules, which is presumably more than necessary. Using atomic symbols for feature structures,<sup>4</sup> an example spurious rule learnt is  $\{V1, V0\} \rightarrow V0\ Det$ . This rule is both implausible, and also overgenerates. For example, it would allow the ungrammatical sentence:

**32** *\*Sam chases the the the cat*

to be generated.

If the sentence is reparsed as before, using a model consisting of just the LP rules, then we have a reduction in the number of rules acquired:

---

<sup>4</sup>V1 stands for a bar one verbal feature structure and V0 for a bar zero verbal feature structure.

```

39 Parse+>> Sam chases the happy cat
learning
9 rule(s) acquired.
85 parse(s)
11777140 msec CPU, 12798000 msec elapsed

```

Similarly, if the sentence is re-parsed as before but using a model consisting of just the semantic types, an even greater reduction in rules acquired occurs:

```

45 Parse+>> Sam chases the happy cat
learning
6 rule(s) acquired.
8 parse(s)
165420 msec CPU, 181000 msec elapsed

```

In both cases, the model is not constraining enough and implausible rules are acquired. If both LP rules and semantic types are used together, the following result is found:

```

29 Parse+>> Sam chases the happy cat
learning
1 rule(s) acquired.
1 parse(s)
5940 msec CPU, 7000 msec elapsed

```

The parse generated is:

```

30 Parse+>> !*pareses*
((('S1'
  ((|Sam|)
    (('VP'
      ((|chases|)
        (('NP1' ((|the|)
          (('*binary25158' ((|happy|)
            (|cat|))))))))))

```

The names of the rules used in the parse label the nodes of the tree and the newly acquired rule (\*binary25158) in full is:

$$\left\{ \begin{bmatrix} N + \\ V - \\ BAR \{1,2\} \\ DET - \\ PER 3 \\ PLU - \\ NTYPE COUNT \end{bmatrix}, \begin{bmatrix} N + \\ V + \\ BAR \{1,2\} \\ DET - \\ ADV - \end{bmatrix} \right\} \rightarrow \begin{bmatrix} N + \\ V + \\ BAR 1 \\ DET - \\ ADV - \end{bmatrix} \begin{bmatrix} N + \\ V - \\ BAR \{1,2\} \\ DET - \\ PER 3 \\ PLU - \\ NTYPE COUNT \end{bmatrix}$$

This is paraphrased into an atomic phrase structure rule as:

$$\{AP, NP, Adj, N1\} \rightarrow Adj N1$$

This rule leads to overgeneration as the model cannot refine the disjunctive LHS into a single non-disjunctive category. Chapter five shows how data driven learning can refine such a rule. Note that the search space explored (as roughly measured by the time elapsed) is much less than that of the previous examples.

If the previous sentence is reparsed as before, using both LP rules and semantic types, but this time also using the Head Feature Convention, then the rule acquired has a slightly different LHS:

$$\left\{ \begin{bmatrix} N + \\ V - \\ BAR \{1,2\} \\ DET - \\ PER 3 \\ PLU - \end{bmatrix}, \begin{bmatrix} N + \\ V + \\ BAR \{1,2\} \\ DET - \\ ADV - \end{bmatrix} \right\} \rightarrow \begin{bmatrix} N + \\ V + \\ BAR 1 \\ DET - \\ ADV - \end{bmatrix} \begin{bmatrix} N + \\ V - \\ BAR \{1,2\} \\ DET - \\ PER 3 \\ PLU - \\ NTYPE COUNT \end{bmatrix}$$

The NTYPE feature is not in the set of head features, and hence it does not obey the HFC and so no longer appears in the rule's LHS.

The system is capable of learning ambiguous attachments. For example, prepositional phrases are traditionally thought to attach ambiguously. The original grammar  $G$  lacks rules to attach the rule PP and these rules will need to be learnt. If the system tries to learn rules for the sentence:

using the full model (LP rules, types, and the HFC), the binary super rule and the original grammar (which contains no rules for PP-attachment), the following result occurs:

```
17 Parse+>> Sam chases the cat down the road
learning
8 rule(s) acquired.
13 parse(s)
189290 msec CPU, 226000 msec elapsed
```

One of these rules attaches the PP to the N1, another to the NP, and another to the VP. The PP is not attached to the S due to LP4 (which says that prepositional phrases do not attach to sentences). The other 5 rules<sup>5</sup> are implausible and caused by the model being underconstraining. Chapter five shows how model incompleteness can be overcome inductively. Note that if the original grammar had a rule that attached PPs, then no undergeneration would take place. Consequently, other ambiguous attachments would not be learnt. Likewise, if the system learnt about PP-attachment in the context of a sentence such as:

**34** *Sam chases down the road the cat*

then the system would not be able to learn about PP-attachment to NPs. This therefore shows that for learning about ambiguity, the system is dependent upon the original grammar and the ordering of the training material.

An advantage of model-based learning is that ungrammatical sentences can be recognised and not lead to a performance grammar being acquired. For example, using the full model and the binary super rule:

```
52 Parse+>> Sam chases happy the cat
learning
0 parse(s)
3010 msec CPU, 3000 msec elapsed
```

No rules are acquired (due to semantic type checking). Model incompleteness will,

---

<sup>5</sup>Some of these implausible rules are shown on page 89.

however, lead to some ungrammatical sentences having parses using learnt rules that the model could not reject:

```
68 Parse+>> Sam chases the cat happy
learning
1 rule(s) acquired.
1 parse(s)
69 Parse+>> !*parses*
(((('S1'
  ((|Sam|)
    (('VP'
      ((|chases|)
        (('NP1' ((|the|)
          (('*binary1288' ((|cat|)
            (|happy|)))))))))))
```

This implausible rule ( $\{N1, NP, Adj, AP\} \rightarrow N1 Adj$ ) is learnt due to a lack of linear precedence information regarding the ordering of adjectives and nominals.

Throughout this section, all of the examples used just the binary super rule. The system is also capable of learning unary rules. Here the model is as before (LP rules, types and Head Feature Convention), but this time both binary and unary super rules are used:

```
83 Parse+>> the happy cat
learning
8 rule(s) acquired.
15 parse(s)
318060 msec CPU, 377000 msec elapsed
```

As expected, the search space has greatly expanded. Most of the extra rules learnt are chains of unary rules, up to the maximum bar level in depth (as, for example  $AP \rightarrow Adj, A\beta \rightarrow AP$ ). When using just the binary super rule (with the same learning configuration as before), only the single rule is learnt:

```

10 Parse+>> the happy cat
learning
1 rule(s) acquired.
1 parse(s)
1950 msec CPU, 3000 msec elapsed
11 Parse+>> !*parses*
((('NP1' ((|the|) ('*binary3072' ((|happy|) (|cat|)))))))

```

This is the same as rule \*binary25158 previously learnt.

## 4.6 Discussion

This chapter explained what a model of grammaticality is, and went on to show how a grammar could be extended using such a model. There are several advantages to using a model when learning grammar:

- Plausible rules can be learnt.
- Ungrammatical strings can be recognised for what they are.
- Expressing linguistic generalisations concisely within a model removes the need to discover such generalisations inductively, using large quantities of data.
- Languages can be identified in the limit using a model.

We saw from the examples how plausible rules can be learnt. As well as helping acquire plausible rules, the model also supplied a justification of the rules. That is, each rule learnt satisfied the model. Inductive approaches provide no such support for the existence of any given rule.

Linked with the previous point is the fact that if the model rejects all of the rules for a string, that string is ungrammatical. The model justifies the rejection of a string and this justification can then serve as the basis for string correction. Hence, the model doubles-up as both a yardstick of grammaticality, and also as a yardstick of ungrammaticality.

Model-based learning is knowledge-intensive, and not data-intensive. This means that large quantities of data are not necessary to learn a grammar using a model-



based learner. Hence, all things being equal, a model-based learner will learn a ‘reasonable’ grammar faster than a data-driven learner.

As has been previously mentioned, a model of grammaticality acts as an informant and enables the learner to eventually learn the language in question. Inductive approaches, without an informant, cannot identify the language and hence will always be limited solutions to the problem of language learning.

Various problems face model-based learners:

- Model incompleteness undermines learning.
- The model may be inconsistent and hence may lead to an implausible grammar being learnt.
- The model may contain intractable principles that take exponential or more time.

We saw in this chapter how model incompleteness undermines learning. That is, the system learnt more rules than was necessary for the PP-example and acquired implausible rules for an ungrammatical sentence. In the next chapter a solution to the problem of model incompleteness is considered.

Ensuring consistency faces any designer of a model-based grammar learners. For example, Fong found that his (GB-based) model of grammaticality would sometimes unexpectedly fail to prove a previously proved sentence after a principle was changed [36, p.209]. There is no absolute solution to this problem, other than using intuition and hoping that testing uncovers inconsistencies.

Intractable principles can be dealt with either by placing a resource bound upon the principle, or by removing the principle from the model. For example, Fong found that some GB principles of grammaticality (such as VP-adjunction) led to parser non-termination [36, p.49]. He placed resource bounds upon these principles, thereby introducing incompleteness into his model. Within the Grammar Garden, either solution to the intractable principle problem can be adopted. The data-driven learner would then be expected to compensate for the resulting incompleteness.

In the next chapter, data-driven learning will be presented.

## Chapter 5

# Data-Driven Learning

### 5.1 Introduction

This chapter explains what is meant by *data-driven* (or inductive, or stochastic) grammar learning, presents an example of a data-driven grammar learner, demonstrates some of the characteristics of this learner, and finally discusses data-driven grammar learning in general.

### 5.2 Data-driven learning in machine learning and in linguistics

Induction has long been a popular learning method in machine learning (for example, [129, 83, 81, 111, 98, 99]). Broadly speaking, when sufficient instances of some concept have been seen, a concept description can then be constructed. The concept description can then be used to classify other instances. For example, having observed the set of instances  $\{a, aa, aaa\}$ , the learner might construct the concept (regular expression)  $a^n$ . Clearly, this describes the instances, along with other, unseen instances, such as  $aaaa$ . Inductive approaches have also been used in linguistics. Harris [48] and others developed procedures to carry out “distributional analysis”; the procedure, when applied to a corpus, derived the rules of grammar from the corpus [73, p.157]. This line of research fell by the wayside. However, more recently, speech recognition researchers have re-adopted inductive grammar construction approaches [76, p.13]. These workers base their work on Shannon’s *Noisy Channel*

*Model* [109, 25]. This can be described as follows. Suppose a sequence of objects  $P$  has been corrupted in some manner, giving the sequence of objects  $W$ . The most likely sequence of objects  $\rho$  can be recovered from  $W$  by hypothesising all possible sequences of  $P$  and selecting the sequence of  $P$  that is most likely. That is,

$$\rho = \operatorname{argmax}_P Pr(P)Pr(W | P) \quad (5.1)$$

$Pr(P)$  is the probability that  $P$  will be present in the channel and  $Pr(W | P)$  is the probability of  $W$  given  $P$ . Argmax finds the argument with the maximal score. Channel characterisation is in reality far too difficult to achieve correctly and so channel approximations such as  $N$ -grams and Stochastic Context Free Grammars are used instead to determine the most likely sequence of events. These approximations are inductively constructed (which constitutes the learning aspect of using the Noisy Channel approach), using training material of some form.

Corpus linguists (for example [40, 112, 46, 120, 6]), have, following the success of the speech recognition community's use of the Noisy Channel Model, also adopted this model. A popular reformulation of the model is to use a *treebank* and a matching algorithm as an approximation of the channel [71]. A treebank is a set of parses for some corpus. This reformulation, borrowing Magerman's terminology, is called *treebank recognition* [76, p.59]. In treebank recognition, broadly speaking, the task is to find a parse  $\rho$  for some sentence that is sufficiently close to some parse  $\phi$  in the treebank. Alternatively, putative parses for some sentence are compared against the treebank, and the putative parse that is closest to some parse in the treebank is deemed to be the 'correct' parse. Because the matching algorithm implements a total function, any parse can be compared against the treebank. Hence, treebank recognition is 'complete': any sentence, no matter how badly formed, will have a closest parse in the treebank. Accepting a parse that is not present in the treebank makes an inductive leap. Given this characteristic, a grammar learner could 'read-off' the newly constructed rules present in the parse that is closest to the parses in the treebank. This is the approach outlined in the next section.

### 5.3 A data-driven grammar learner

Here, an example of a data-driven grammar learner, similar in style to those used by corpus linguists, is presented. This is not intended to be the final statement on inductive grammar learning. Instead, it is demonstrative of inductive grammar learners and intended to show how such learners might benefit from model-based learning.

The first task is to construct the treebank and the second task is to create a tree matching algorithm.

Treebanks can be created either manually, or automatically [105, 117]. Manual treebank construction is very labour intensive and error prone. Automatic treebank creation, which does not suffer from these logistical problems, is preferable. Treebanks can be automatically created by selecting a corpus, parsing that corpus, and then recording the parses produced. To be successful, the grammar used in treebank creation should be able to generate all of the corpus. However, undergeneration will undermine automatic treebank creation. One way of overcoming the problem of undergeneration would be to record, as well as complete parses, local trees produced for sentences in the corpus. Another way of automatically creating a treebank would be to run a grammar in reverse and generate the corpus artificially.<sup>1</sup> An obvious problem with this approach is that the frequencies of various constructs would be unnatural, and hence would not be representative of any natural language.

In this data-driven learner, the treebank is created by parsing a corpus and then recording all local trees generated. We are not interested in dealing with syntactic ambiguity and so there is no need to have a parse selection mechanism that chooses the ‘best’ parse for any given sentence in the corpus. Following the example of Leech [70, 71], we decompose the treebank into a set of mother-daughter-frequency triples. This is designed to compress the treebank and make it more manageable. The mother of the triple is the root of a local tree, the daughter is a category immediately dominated by the mother, and the frequency is the total number of times that that mother-daughter pair have been seen in the treebank. For example, a treebank consisting of the two parses:

---

<sup>1</sup>The ARK Corpus was constructed in this manner [117].

(S (NP Sam) (VP (V laughs)))

and

(S (NP Sam ) (VP (V chases) (NP (Det the) (N cat))))

would be decomposed into the set of triples:

$\langle S, NP, 2 \rangle$

$\langle S, VP, 2 \rangle$

$\langle VP, V, 2 \rangle$

$\langle VP, NP, 1 \rangle$

$\langle NP, Det, 1 \rangle$

$\langle NP, N, 1 \rangle$

Note that decomposing a treebank in this manner throws away some structural information. Also, triples are not produced for local trees immediately dominating lexical material. Adding such triples would introduce problems of sparse statistics (lexical items occur less frequently than syntactic categories). When using a unification-based grammar to create the treebank, the test to determine the frequencies of the triples is category equality. Category  $D$  is equal to category  $D'$  if  $D \subseteq D'$  and  $D' \subseteq D$ .

After creating a set of triples, the next task is to create the matching algorithm. This should give frequently seen trees in the treebank a high score, infrequently seen trees a low score, and unobserved trees a small score. By giving unobserved trees a small score, the matching algorithm makes the ergodic assumption,<sup>2</sup> and hence is complete. The algorithm should also allow frequently observed trees in the treebank (which can be seen as having a high degree of confidence of being ‘correct’) to support other trees. Conversely, trees with a low degree of ‘support’ in the treebank should not support other trees. Given these considerations, the data-driven learner has the following matching algorithm. Let  $\Upsilon$  be a set of triples encoding a treebank. For some triple  $x$  in  $\Upsilon$  of the form  $\langle a, b, f \rangle$ , let  $M(x) = a$ ,

---

<sup>2</sup>The ergodic assumption, roughly speaking, is that all events have a non-zero probability of occurring. Hence, all events might be seen, and none are ruled-out.

$D(x) = b$ , and  $F(x) = f$ . Define the function  $lookup: A \times B \mapsto [0, 1]$ , where  $A$  and  $B$  are categories, as follows:

$$lookup(A, B) = \begin{cases} \sum_{\forall i \in \tau} F(i) / \sum_{\forall j \in \Upsilon} F(j) & \tau \neq \{\} \\ \delta & \text{otherwise} \end{cases} \quad (5.2)$$

Here,  $\tau$  is the set of triples drawn from  $\Upsilon$ , such that each triple  $x$  in  $\tau$  satisfies  $M(x) \sqcup A$  and  $D(x) \sqcup B$ .  $\delta$  is a small value that allows the ergodic assumption to be made. Now, a local tree  $t$  of the form

$$(A_0(A_1 a_1 \dots a_n)(A_2 b_1 \dots b_n) \dots (A_n c_1 \dots c_n)))$$

produced during interleaved parsing and learning is scored as:

$$score(t) = geo\text{-}mean(lookup(A_0, A_1), lookup(A_0, A_2), \dots, lookup(A_0, A_n)) \quad (5.3)$$

Here, the categories  $A_1 \dots A_n$  immediately dominate lexical items. The geometric mean, and not the product, is used to compose scores, thereby avoiding penalising local trees with more daughters over local trees with fewer daughters [74].

Interior trees of the form

$$(A_0(A_1(B_1 \dots))(A_2(B_2 \dots)) \dots (A_n(B_n \dots)))$$

where  $A_1 \dots A_n$  immediately dominate other nonterminals are scored as:

$$\begin{aligned} score(t) = geo\text{-}mean \quad & (lookup(A_0, A_1) * score(A_1), lookup(A_0, A_2) * score(A_2), \\ & \dots, lookup(A_0, A_n) * score(A_n)) \end{aligned} \quad (5.4)$$

Including the scores of dominated material provides contextual support for local trees.

If some  $A_i$  is a disjunctive category, the score then is the maximal score of the local, non-disjunctive trees encoded disjunctively. Taking the maximum is equivalent to an interpretation of disjunction used in fuzzy logic.

Finally, after scoring local trees, a local tree  $t$  of the form

$$(A_0(A_1(B_1 \dots))(A_2(B_2 \dots)) \dots (A_n(B_n \dots)))$$

is judged to be sufficiently similar to previously seen local trees (encoded in the treebank) if:

$$geo\text{-}mean(score(A_0), score(A_1), \dots, score(A_n)) > \omega \quad (5.5)$$

where  $\omega$  is a small value, greater than  $\delta$ . This judgement forms the inductive component of the Grammar Garden’s critic. Only local trees corresponding to instantiations of the super rules are subject to judgement, but because of the need to provide contextual support, all local trees are scored.

In summary, a treebank is collapsed into a set of triples, which are then used by a tree matching algorithm. The algorithm is then used to score local trees produced during interleaved learning and parsing and local trees arising from usage of super rules are subject to inductive criticism. Our data-driven learner is similar to other approaches (for example [70, 46, 120, 94]) based upon treebank recognition.

As an aside, it is sometimes useful to post-process a grammar. Post-processing is aims to reduce a grammar’s overgeneration. This is achieved by refining the grammar. Grammar refinement consists of removing disjuncts from disjunctive categories, removing rules if they have insufficient inductive support and removing rules whose structural support has been undermined.

Removing disjuncts is achieved by multiplying-out rules encoded within the single disjunctive rule and scoring each such rule. Scoring is the same as for local, non-interior trees and the rule with the highest score is selected, replacing the previous disjunctive rule within the grammar. If no single highest scoring rule exists, the disjunctive rule remains within the grammar. For example, the disjunctive rule  $\{A, B\} \rightarrow C D$  would be expanded into  $A \rightarrow C D$  and  $B \rightarrow C D$ . If  $score(A \rightarrow C D) > score(B \rightarrow C D)$  then  $A \rightarrow C D$  would be used to replace  $\{A, B\} \rightarrow C D$ .

If the score of a rule fails to exceed a small value then that rule is removed from the grammar.

Removing rules whose structural support has gone is best shown with an example. Suppose the rule  $A \rightarrow B C$  was learnt in the context of the local tree (A (B (E F)) C). Now, if the rule  $B \rightarrow E F$  was no longer in the grammar<sup>3</sup>, then arguably, the structural support for the rule  $A \rightarrow B C$  is absent, and so the rule should be removed from the grammar. To achieve this, the local tree dominated by a learnt rule is recorded and if any rule within that local tree is no longer within the gram-

---

<sup>3</sup>Other options might be to only remove the rule if it is no longer *useful*. A useful rule is one that can be eventually re-written as a sequence of terminals.

mar, the rule is also removed from the grammar. This process is similar to ensuring consistency in truth maintenance systems

This concludes the description of the data-driven learner. It is now time to look at examples of data-driven learning.

## 5.4 Examples of data-driven grammar learning

This section demonstrates the capabilities of data-driven learning by firstly representing the learning example given in chapter three, showing the effect of gradually increasing the inductive threshold  $\omega$ , secondly by giving an example of rule refinement, and finally, by showing how data-driven learning can improve on the performance of model-driven learning. The grammar and lexicon used in these examples are the same as in the previous chapter.

Prior to learning, the data-driven learner needs a treebank for tree scoring. To this end, the following sentences were parsed:

**35** *Sam chases the cat.*

**36** *The cat chases Sam.*

**37** *The cat down the road chases Sam.*

**38** *\*Sam down the road chases the happy cat.*<sup>4</sup>

Parsing this (tiny) corpus produced 68 triples. Now, suppose the *Sam chases the happy cat* sentence is processed, using just data-driven learning, with an  $\omega$  value of 0.35. As before, the grammar cannot generate this sentence, and so learning takes place:

```
165 Parse+>> Sam chases the happy cat
learning
8 rule(s) acquired.
63 parse(s)
10083090 msec CPU, 11104000 msec elapsed
```

---

<sup>4</sup>This ungrammatical sentence is introduced to show that the data-driven learner cannot validate the grammaticality of the training set.



Here, the  $\omega$  value is too small, and little data-driven criticism takes place. Repeating this experiment, but with a higher  $\omega$  value of 0.40 produces the following result:

```
155 Parse+>> Sam chases the happy cat
learning
4 rule(s) acquired.
7 parse(s)
201300 msec CPU, 232000 msec elapsed
```

Increasing the threshold has resulted in super rule instantiations being rejected. Using atomic symbols as a notational convenience, the four rules learnt are:

$$\{A1, A2, N1, N2\} \rightarrow A1 \ N1$$

$$\{V0, V1\} \rightarrow V0 \ Det$$

$$\{A0, A1\} \rightarrow Det \ A0$$

$$\{N2, N3, V0, V1\} \rightarrow N2 \ V0$$

As it turns out, these four rules are all equally scored and so are indistinguishable from each other. As such, the data-driven learner cannot select the desired rule from the undesired set of rules. Still, without LP rules, types, or a Head Feature Convention, it has managed to eliminate four other implausible rules.

This has shown examples of how data-driven learning compares with model-based learning. However, data-driven learning can compensate for incompleteness within the model of grammaticality. Consider rule LHSs. As can be seen from the previous learnt rules, they all contain disjunctions. This is because the rule constructor usually cannot determine what the LHS category should be. However, data-driven learning is capable of refining rules. Suppose the model-based learner acquired the rule:

$$\{A1, N2, A2, N1\} \rightarrow A1 \ N1$$

This rule, being disjunctive, needs to be refined. The data-driven learner can refine this rule by monitoring its usage in parse trees. For example, when parsing the following sentence:

```
167 Parse+>> Sam chases the happy happy cat
1 parse(s)
6100 msec CPU, 7000 msec elapsed
```

the system could record the triples produced. This exposure to data causes an increase in the frequency of the mother-daughter pairs  $\langle N1, A1 \rangle$  and  $\langle N1, N1 \rangle$  over and above that of the frequency of the mother-daughter pairs (such as  $\langle NP, N1 \rangle$ ) which support other categories as candidates for the LHS. The system could then determine how the rule is used:

```
168 Parse+>> !(refine-grammar)
Refining and deleting rules ...
Refining 4 rules encoded in *binary510746 score: 0.14390989949130545
rule is N1 -> A1 N1
```

That is, the data-driven learner has refined the learnt rule, throwing away the extra disjuncts, and giving the rule:

$$N1 \rightarrow A1 N1$$

This example of rule refinement is something that the model-based learner, through incompleteness, could not achieve.

Finally, if we return to the PP-problem in the last chapter, we saw how the model-based learner acquired more rules than necessary:

```
17 Parse+>> Sam chases the cat down the road
learning
8 rule(s) acquired.
13 parse(s)
189290 msec CPU, 226000 msec elapsed
```

Now, if data-driven learning was also used, then some of these implausible rules could be rejected. The following table shows the number of rules rejected, when using model-based learning and data-driven learning, for the PP-problem, against a varying  $\omega$  value:

$\omega$ value	Number of rules acquired
0.010	8
0.040	7
0.050	6
0.060	5
0.062	4
0.070	3
0.080	2
0.090	1
0.100	0

Hence, data-driven learning can reduce the number of spurious rules learnt. However, being inductive, it can also throw away rules that are not spurious. For example, when  $\omega = 0.070$ , the rules learnt are:

$$\{V0, V1, P3, P2, N3\} \rightarrow V0 \{P3, N3, P2\}$$

$$\{V1, V2, P2, P3\} \rightarrow V1 P2$$

$$\{N2, N3, P2, P3\} \rightarrow N2 P2$$

When  $\omega = 0.080$ , the first of these rules is thrown away. Whilst this rejects a rule such as:

$$\{V1, V0, N3\} \rightarrow V0 N3$$

it also rejects a rule that attaches the PP to the verb. As can be seen, the relationship between  $\omega$  and the rules rejected is much less clear than for the model-based learner. However, it is far easier in data-driven learning to vary the aggression of the critic than with model-based learning. In data-driven learning, all that needs to be changed is the  $\omega$  value: higher for fewer rules being learnt, lower for more rules being learnt. In model-based learning, to increase the rejection level means extending the model of grammaticality. This entails knowledge engineering, which is obviously labour intensive.

In sum, we saw how data-driven learning works, and how it can interact with model-based learning. As should be apparent, neither learning style in isolation is ideal, but using both together helps achieve a better solution overall.

## 5.5 Discussion

This chapter explained what is meant by data-driven learning and went on to show how a grammar could be extended using this learning style. The chapter also showed how data-driven learning can be used in conjunction with model-based learning.

There are two advantages to using data-driven learning:

- The learner is complete.
- There is little logistical effort required, other than the creation of the treebank.

As was previously stated, ‘completeness’ here means that the learner can always make a decision. In the context of a learner using a treebank and tree matching algorithm, this means always returning a non-zero score for any local tree being matched. Advocates of data-driven grammar learning often cite this advantage as a rationale for their work. This is because they value the ability of inductively constructed grammars always to provide an analysis for some sentence, no matter how ungrammatical that sentence might be. As should be clear from the stance taken in this work, not everyone (including the author) views this as an advantage. Some situations, such as message understanding, might always require some syntactic analysis. Other applications, such as handwriting recognition<sup>5</sup> of cheques, might demand that the system at times rejects input that is unrecognisable.

The second advantage means that knowledge engineering is kept to a minimum. Linguists are not required to formulate grammatical models within the data-driven approach to grammar learning. It could therefore be argued that data-driven approaches are ‘theory-neutral’ and hence unaffected by reformulations (or abandonment) of linguistic principles.

Data-driven learners also have weaknesses:

- They cannot identify in the limit natural languages.
- They usually approximate the Noisy Channel Model and hence make mistakes over and above their theoretical limitations.

---

<sup>5</sup>Grammars have also been used to recognise handwriting (for example [13]).

- There never will be enough data to train upon and hence the grammars will be undertrained.
- Data-driven learners cannot distinguish rare constructs from ungrammatical sentences.

The first weakness follows from the work of Gold and was considered in chapter two.

Approximate formulations of the Noisy Channel Model include using inadequate formalisms, using noisy training data, and using decision theories that do not always have the desired behaviour.

Examples of using inadequate formalisms can be seen in the work of Garside *et al.* [40], or in the use Hidden Markov Models in speech recognisers (for example [57]). These are all finite state technologies, which cannot capture all natural language constructs. Apart from theoretical objections, there is empirical support of the inadequacy of finite state technologies. Lari and Young, in their experiment of learning the palindrome language (which is context free), found that using a context free grammar produced better results than when using a Hidden Markov Model [69].

Noise undermines the ability of the data-driven learner to make correct decisions. In the context of a data-driven learner using a manually constructed treebank, variability of analyses means that regularities being sought cannot always be found. This problem has been noted in the literature (for example, Brill *et al.* comment that in the Penn Treebank, on average 3.2% of the words are mis-tagged [9]; Black *et al.* comment that the treebank that they use contains 2.5% noise [6, p.194]).

A *decision theory* is one that helps the learner decide whether to accept an hypothesis, or reject that hypothesis. Inductive grammar learners typically use a statistical decision theory. However, as Carroll shows, these decision theories do not always give the desired results [16, p.133]. For example, associating probabilities with context free rules means that the decision theory cannot arbitrate between different derivation sequences. In the context of grammar learning, a differing derivation sequence might correspond to a competing set of rules for some sentence. Hence, such a decision theory would not always be able to decide which set of rules to prefer. Regarding the data-driven learner, using the context of a local tree can be seen as an attempt at overcoming this problem. Note that taking the

geometric mean means that the scores given to local trees do not have probabilistic interpretations.

Returning to the list of weaknesses of data-driven learners, it is clear that, assuming natural languages are infinite, there will never be enough data available to fully train an inductively constructed language model. However, researchers can approximate an infinite size training set with a training set that is suitably large. Unfortunately, ‘suitably large’ can be very large indeed. For example, Church and Mercer show that to obtain reliable information about the adjective ‘strong’ requires at least 46 million words [25]. Of course, this is an upper bound, and less data could be used, assuming that smoothing techniques are employed to deal with the resulting sparse statistics [39]. Smoothing approaches, however, can only estimate the underestimated parameters of an inductively constructed language model. Therefore, in practise, the size of the training sets required will pose such a formidable computational task that the resulting grammar, even with smoothing, will be undertrained.

As data-driven learners base their concept of grammaticality upon frequency, the learner cannot distinguish between rare constructs and ungrammatical constructs. Rare constructs will have a low probability (by definition). Ungrammatical constructs will also (hopefully) have a low probability. Therefore, if a low probability is taken to mean an ungrammatical construct, then correction of such a construct would mean that grammatically well-formed sentences would be incorrectly corrected. This is suboptimal behaviour and undermines the performance of a sentence correcting device.

All of these weaknesses with data-driven learning can be tackled by also using model-driven learning. Chapter seven considers the link between these two learning styles in detail.

The previous two chapters demonstrated various aspects of the learners. In the next chapter, the system is evaluated with naturally occurring language.

## Chapter 6

# Evaluation

### 6.1 Introduction

The previous two chapters demonstrated capabilities of the learning system. This chapter evaluates the system in terms of the success criteria introduced in chapter two and principally aims to demonstrate that combining data-driven and model-based learning produces qualitatively better grammars than are produced when using either learning style in isolation.<sup>1</sup>

Broadly speaking, there are two approaches to evaluating machine learning systems. They can be either *formally* or *empirically* evaluated. A formal evaluation will prove the system meeting the success criteria, whilst an empirical evaluation will only show the system meeting the success criteria. In chapter two, one such formal analysis (Gold’s Identification in the Limit [42]) was introduced. Another, more recent formal approach is *Probably-Approximately-Correct Learning* (PAC learning) [123]. The has-it or has-it-not identified the language nature of Gold’s approach is replaced with a how-well-has-it identified the language framework. PAC learnability is therefore better suited than Gold’s approach for many machine learning systems. Yet another formal approach is computational complexity theory, which gives characterisations of time and space requirements of solutions to various classes of problems [5]. Unfortunately, none of these formal approaches are suitable for evaluating the learning system. Gold’s approach is too coarse and says nothing about

---

<sup>1</sup>Some of these experiments have also been published in two papers [92, 91].

approximations [84, p.3]. PAC learnability talks about approximations, but the theory is not sufficiently advanced to deal with learning natural language grammars [14, 66]. Complexity theory considers worst case behaviour and does not address domain specific issues such as parse plausibility or undergeneration. Indeed, Kibler and Langley comment that “...many learning algorithms remain too complex for formal analysis. In such cases, empirical studies of the behaviour of these algorithms must retain a central role.” [65]. Hence, the approach taken when evaluating the system in this thesis is empirical.

Related work on empirically analysing grammars tend to concentrate upon measuring ‘correctness’ (plausibility) of parses and/or measuring overgeneration. Plausibility is usually measured either qualitatively (by manually inspecting parses, for example [11, 12]) or quantitatively (by tree-matching algorithms, for example [49, 6]). Qualitative measurement is labour intensive and hence quantitative methods are (from a logistical perspective) preferable. Overgeneration is usually quantitatively measured (for example [69, 15, 12]) in terms of how random the language generated by the grammar is. The more random the language, the greater the grammar’s overgeneration. Few approaches consider undergeneration, given the ergodic assumption made by most inductively constructed grammars, which eliminates undergeneration. An ideal approach would consider all three aspects of a grammar (undergeneration, overgeneration, and plausibility), not just any one in isolation. This would counter arguments such as success at dealing with undergeneration being due to excessive overgeneration. Unfortunately, few systems are evaluated in terms of all three aspects of grammaticality. This thesis uses all three aspects of grammaticality.

The experiments in this chapter follow the same outline:

- Run the configured learner over the training material and produce a grammar.
- Evaluate that grammar using a set of metrics.

In general, there are many aspects of the learner that could be tested and so for logistical reasons, only those relating to the success criteria outlined in chapter two will be considered. So, the training material (for example) will be kept constant, but the learner’s configuration will vary.

The rest of this chapter is as follows. Section 6.2 outlines the metrics used;



section 6.3 describes a set of experiments evaluating the learning system and finally, section 6.4 discusses what has been revealed about the learning system.

## 6.2 Metrics

This section presents three quantitative metrics measuring grammar quality: a metric testing a grammar's undergeneration, a metric testing a grammar's overgeneration, and finally, a metric testing for parse plausibility.

### 6.2.1 Measuring undergeneration

Since undergeneration is defined as a grammar's inability to generate a grammatical sentence, an obvious way to measure this would be to construct a set of grammatical sentences and then determine how many of these sentences are generated by the grammar. The more of these sentences generated, the lower the undergeneration of a grammar. Grammatical sentences can be found in a corpus of naturally occurring language.

### 6.2.2 Measuring overgeneration

Testing for overgeneration is similar to testing for undergeneration. A set of ungrammatical strings should be selected and we should determine how many of these are generated by a grammar. The fewer strings generated by the grammar, the lower the grammar's overgeneration. It is harder to find a set of ungrammatical strings than it is to find a set of grammatical sentences. One possible way to locate strings would be by concatenating different numbers of randomly chosen terminals together to generate a set of strings of any length. Such a set would be *largely* ungrammatical on the grounds that natural languages are predictable. Clearly, random generation of strings results in a language that is not predictable. Other researchers (for example [69, 15, 12, 62]) use this idea indirectly and measure overgeneration as the *entropy* of the language generated by the grammar. Entropy is a measure of how much information is produced by (say) a word in a text. If the words are replaced by bits in an optimal manner, then the entropy  $H$  is the average number of bits required per word: the higher the entropy, the greater the ungrammaticality of the

set of strings. For example, Shannon estimated the entropy of encoding letters in English roughly to be 2.3 bits per letter [109]. A problem with using entropy is that the figure gives no insight into how a grammar overgenerates. By measuring overgeneration directly in terms of generating ungrammatical strings, it is easier both to locate sources of overgeneration and also to see what the figure for overgeneration means. This work therefore uses the random string generation approach to measuring overgeneration.

### 6.2.3 Plausibility

It is difficult to determine if a parse for some sentence is plausible. For example, Harrison *et al* reported an experiment involving the comparison of manually produced parses for 50 sentences taken from the Brown Corpus [49]. The results revealed that there was little common structure between the parses, reflecting a diversity of approaches to punctuation, empty categories, and so on. Hence, what is plausible for one person is not plausible for another. One popular avoidance of this problem with manual plausibility determination is to define parse plausibility as conformity to a benchmark parse. The greater the deviation of the test parse from the benchmark parse of the same sentence, the less plausible the test parse.

The approach in this work is based loosely upon the work of Harrison *et al*<sup>2</sup> and the matching algorithm consists of the following steps:

1. Normalise the test parse to use the same labelling scheme as the benchmark parse.
2. Flatten both test parse and benchmark parse into the lists  $\tau$  and  $\beta$  by a preorder walk.
3. Starting from the head of  $\tau$ , find the longest list that is in  $\tau$  that is also in  $\beta$ . Remove this list from  $\tau$ . Repeat this, removing the longest list again, until either  $\tau$  is empty, or no such list is common to both  $\tau$  and  $\beta$ .
4. The match between the test parse and the benchmark parse is the arithmetic

---

<sup>2</sup>The chief difference between our matching algorithm and their approach is that we match against the entire tree, whilst Harrison *et al* only match against the bracketing of the tree.

mean of the list lengths of the lists found in step 3 divided by the list length of  $\beta$ . A score of 1 is a perfect match and a score of 0 is a perfect mismatch.

The test parse has to be normalised so that like is compared with like. This is particularly true when the grammar formalisms and the assumptions behind the analyses differ. For example, grammars learnt in this thesis are unification-based and assign steep parses. The benchmark parses use atomic labels and are shallow [40, 71]. Normalisation is performed by mapping feature structures to atomic categories. Categories with bar levels greater than one are mapped to phrasal atomic categories, thereby flattening steep parses. By matching lists, and not trees, some structural information is thrown away. Again, this helps in the normalisation process, at the cost of making the match only approximate.

For example, if  $\tau$  was the list (a b c d) and  $\beta$  the list (c a b c), then the first longest sublist common to both would be (a b c). Removing this list from  $\tau$  results in  $\tau$  becoming the list (d). As there are now no lists common to both  $\tau$  and  $\beta$ , matching halts, with the matching lists being {(a b c)}. The closeness score would then be 3/4.

Although the matching algorithm is simple, it does give adequate results. For example, the University of Pennsylvania Treebank benchmark parse:

```
(S (S (NP MISS X) (VP WAS BEST))
  (SBAR WHEN (S SHE (VP NEED (V BE (ADJP TOO PROBING))))))
```

when matched against the test parses:

```
(S (NP-S MISS X) (VP WAS BEST)
  (S WHEN SHE (VP NEED (V BE (ADJP TOO PROBING))))
(S (S (NP MISS X) (VP WAS BEST))
  (S (WHEN (S (NP SHE) (VP NEED BE (ADJP TOO PROBING))))))
(ADJP (S (ADJP (NP-S MISS (VP X WAS)) BEST) WHEN)
  (NP-S SHE (NP-S (NP-S NEED BE TOO) PROBING)))
```

gives a highest match with the first test parse. The lowest match is with the last parse (which was randomly generated).

In practise, many parses will be produced for a sentence and so the first  $k$  ( $k = 10$ ) parses are sampled and matched with the benchmark parse. We record the

score for the best matching tree. Note that if all of the parses that a grammar could generate for some sentence were considered, then the plausibility matching would benefit from overgeneration. That is, if some grammar generates all possible parses, then there would be one that matches exactly with the benchmark parse. However, sampling only  $k$  parses ensures that the matching only looks at a subset of these parses. This means that on average, assuming that most sentences have more than  $k$  possible parses, then the effects of overgeneration can be minimised.

### 6.2.4 Comments

Given these metrics, grammar  $A$  is of higher quality than grammar  $B$  if  $A$  undergenerates and overgenerates less than grammar  $B$  and assigns more plausible parses than  $B$ .

None of these metrics are exact measurements, due to the fact that they all sample finite subsets of the infinite languages that each grammar can generate. There is a risk that some unrepresentative subset of this language is sampled, thus undermining the results. For example, the supposed ungrammatical strings could in reality be sentences. However, by selecting naturally occurring sentences when measuring undergeneration, by randomly generating strings when measuring overgeneration, and by using benchmark parses that are manually produced so as to be plausible when measuring plausibility, the representativeness of each set of data is enhanced. This is similar to stratified sampling techniques used by statisticians trying to overcome unrepresentativeness when dealing with large sample spaces.

## 6.3 Experiments

This section describes a series of experiments that determines how well various configurations of the learner meet the success criteria. It also describes an experiment showing the convergence rate of the system. Convergence is defined as the system identifying the language; it is worth considering, given the fact that a system that converges rapidly is preferable to one that converges less rapidly, all things being equal.

A manually constructed grammar,  $G1$ , consisting of 97 unification-based rules

was used throughout all experiments and extended by various styles of learning.<sup>3</sup> A typical rule from *G1* is:

$$\left[ \begin{array}{c} N - \\ V + \\ BAR\ 2 \\ MINOR\ NONE \\ PLU\ \boxed{6} \\ VFORM\ \boxed{17} \end{array} \right] \rightarrow \left[ \begin{array}{c} N + \\ V - \\ BAR\ 2 \\ MINOR\ NONE \\ PLU\ \boxed{6} \\ CONJ - \end{array} \right] \left[ \begin{array}{c} N - \\ V + \\ BAR\ 1 \\ MINOR\ NONE \\ PLU\ \boxed{6} \\ VFORM\ \boxed{17} \\ CONJ - \end{array} \right]$$

(which can be paraphrased, using atomic symbols, by the rule  $S \rightarrow NP\ VP$ ).

The learning of grammars requires data and in keeping with other researchers (for example [12, 17, 2]), the Spoken English Corpus (SEC) was chosen as a source of training and testing material [71].<sup>4</sup> The SEC consists of *c.* 50,000 words of prepared monologues broadcast over the radio. An advantage of using the SEC is that it is lexically tagged (using the CLAWS2 tagset [6]) and manually parsed (using the UCREL parsing scheme [40]), saving on the need to construct a suitable lexicon or to construct a set of benchmark parses for plausibility evaluation. A typical entry in the SEC is:

```
[N It_PPH1 N]
[V 's_VBZ [N a_AT1 useful_JJ reminder_NN1
           [Fn that_CST
             [N some_DD scientists_NN2 N]
             [V find_VV0 [N Don_NP1 Cupitt_NP1 N]
             unscientific_JJ V]Fn]N]V] ._.
```

Here, the sentence:

**39** *It's a useful reminder that some scientists find Don Cupitt unscientific*

is shown tagged (for example, the word **reminder** has a tag NN1) and parsed (indicated by the square brackets and their adjacent phrasal categories).

<sup>3</sup>The grammar *G1* was kindly supplied by Ted Briscoe (University of Cambridge). A listing of *G1* appears in appendix B. The lexicon of *G1* is given in appendix A.

<sup>4</sup>Access to the SEC was kindly given by Eric Atwell (Leeds University).

As a resource for evaluation, the SEC is quite good. However, in common with all automatically tagged corpora, some of the words in the SEC are mistagged. The SEC sentences are mostly grammatical, given that they have been edited in preparation for broadcast. However, the manual parses in the SEC are uneven in quality. Because of this quality problem, those manual parses used in plausibility evaluation had, in some cases, to be edited. Mostly this editing consisted of adding a sentence root node to the SEC parse trees. Because of the mistagging and the uneven quality of the parses, the experiments will only reveal approximate results of the system. In particular, the plausibility results will only allow relative system performance, and not allow reliable comparison with other systems that use the SEC as data.

All punctuation was removed from the SEC sentences used in the experiments. This was because it was difficult to determine when punctuation was used syntactically (for example in “apples, pears and bread baskets”) or when it was used textually (for example in “John laughed: Bill tickled him”).<sup>5</sup> A grammar, if it is to be a theory of syntax, needs to distinguish between these two uses of punctuation. There has been work on computationally dealing with punctuation [87, 17] and it would be interesting to see, as future work, if punctuation has any impact upon the quality of learnt grammars.

Prior to learning, the 60 shortest<sup>6</sup> sentences were selected from the SEC without regard to how syntactically well-formed they were and set aside as the set of sentences *Train*. These were used as training material. A further (different) 60 sentences were set aside, also selected without regard to syntactic well-formedness, as the set *Test*. These were used to measure undergeneration. A small number of sentences (less than 20), also different from *Train* or *Test*, were also selected (called *Pretrain*) to pretrain the data-driven learner. Pretraining is necessary in order to give an initial estimate of the language model [112]. 15 sentences from *Test* were selected (called *Yardstick*) and each sentence was paired with its associated manual parse taken from the SEC treebank. A further 15 different sentences and their man-

---

<sup>5</sup>The examples on which these are based were supplied by Ted Briscoe (personal communication).

<sup>6</sup>Short sentences were used simply to reduce the computation involved with learning. Other learners, such as DACS [85] also use short sentences.

ual parses (called *Plausible*) were also selected from the treebank. The sentences in *Yarstick* were selected such that they could all be generated by  $G_1$ , whilst the sentences in *Plausible* each required at least one learnt rule in order to be generated. Both of these sets of sentence-parse pairs were used to determine plausibility. *Yarstick* served as the base for comparison with *Plausible*. Finally, 100 strings of length 6 (called *Random*) were randomly generated using grammar  $G_1$ 's lexicon. These strings were used to measure overgeneration.<sup>7</sup> Before proceeding, it is necessary to justify the size of the data sets used in the experiments. By comparison with other researchers, these sets are small, and it could therefore be argued that system evaluation using this amount of data is unconvincing. In other data-driven learning approaches, researchers need to use larger amounts of data. As a consequence of the learning style they use, if they are to achieve a reasonable performance from their technologies, they need to use large data sets. However, evaluation in this thesis is not concerned with measuring how well the data-driven or the model-based learners perform in the limit. Evaluation is concerned instead with determining if using data-driven learning and model-based learning together is better than using either learning style in isolation. Hence, all that matters is that a difference is observed, and there is no need to compare fully trained learners at all. In fact, it could equally be argued that hoping for full convergence is an incoherent idea, given that natural languages are infinite in size, and training sets, being finite, will never be enough. Clearly however, in evaluation, the data sets need to be sufficiently large for any effects to be noticed. As it turns out, good estimates, based upon using only small data sets, can be obtained. *Chernoff bounds* [19] tell us the probable rate of convergence of estimating a variable to the true value of that variable: the probability that the estimate is inaccurate goes to 0 exponentially fast as the size of the data set increases. That is, more is better, but much more is not much better. Using ever larger data sets brings diminishing returns. For example, Brill [8, p.72] explored the effect of training set size on the accuracy of his lexical tagger and found the following behaviour:

---

<sup>7</sup>Listing of *Test*, *Train*, *Bad*, *Yarstick* and *Plausible* appear in appendix D.

Training size (sentences)	Accuracy (%)
1000	90.5
2000	91.7
4000	92.2

That is, quadrupling the number of sentences only increases accuracy by 1.7%. In sum, small (but not tiny) data sets can be used, assuming that any conclusions made based upon these sets do not relate to absolute system performance, but only to relative performance. It is believed that the size of the data sets used in these experiments, though small, are sufficient for showing some of the differences between data-driven and model-based learning. The confidence level results in §6.3.3 support this belief.

Learning, as outlined in chapter three, is computationally very expensive and so resource bounds were placed upon the learners. These bounds were to stop learning when either  $n$  parses or  $m$  edges had been created by the chart parser for some sentence ( $n = 1, m = 3000$ ). Increasing  $n$  leads to more rules being learnt and hence increases the likelihood of finding highly plausible parses (i. e. those that match the benchmark parses exactly). The motivation for the edge limit follows from others who suggest that ungrammaticality might be related to an excessive number of edges being generated for some string [74, 75, 20]. In effect, the chart parser spends a lot of time fruitlessly searching for parses that may not exist. Resource bounds make learning incomplete, but if they are kept constant across all the learning configurations, any incompleteness effects will be factored out. However, incompleteness does mean that all of the evaluation results will be systematically underestimated.

The learner was configured in three distinct ways:

Configuration	Parameters	Grammar produced
A	Data-driven learning only	$G_2$
B	Model-based learning only	$G_3$
C	Data-driven learning and model-based learning	$G_4$

The model-based learner's model consisted of 4 LP rules, 32 semantic types and a Head Feature Convention. In configurations A and C, the system uses *Pretrain* to



get initial frequencies of mother-daughter pairs. All configurations of the learner used X-bar syntax, as this is such a necessary aspect of rule construction.

Each configuration was run over *Train*, causing learning to take place. Using *Test*, the learnt grammars were evaluated with respect to the three metrics. The following table shows the size (in terms of the number of disjunctive and, after multiplying-out, non-disjunctive rules) of the various grammars learnt:

Grammar	Size (disjunctive rules)	Size (non-disjunctive rules)
<i>G2</i>	129	392
<i>G3</i>	128	345
<i>G4</i>	129	385

Appendix C presents, in paraphrased form, the grammars *G2*, *G3* and *G4*.

Each of the grammars was then evaluated as follows.

### 6.3.1 Undergeneration

The following table shows the percentage of sentences in *Test* parsed by all of the grammars:

Grammar	Percentage generated
<i>G1</i>	26.7
<i>G2</i>	75.0
<i>G3</i>	65.0
<i>G4</i>	75.0

As can be seen, learning reduces *G1*'s undergeneration. Data-driven learning (producing *G2*) and combined learning (producing *G4*) jointly reduce undergeneration most.

### 6.3.2 Overgeneration

The following table shows the percentage of sentences in *Random* parsed by all of the grammars:

Grammar	Percentage generated
$G1$	7.0
$G2$	38.0
$G3$	30.0
$G4$	38.0

This time, the reverse has been found: learning increases overgeneration. However, the increase in overgeneration is less than the reduction in undergeneration.

### 6.3.3 Plausibility results

The following table shows plausibility results after matching grammar  $G1$  using *Yardstick*:<sup>8</sup>

Sentence	Score
Y1	0.083
Y2	0.133
Y3	0.092
Y4	0.075
Y5	0.191
Y6	0.079
Y7	0.108
Y8	0.105
Y9	0.088
Y10	0.079
Y11	0.102
Y12	0.174
Y13	0.094
Y14	0.090
Y15	0.053

The arithmetic mean and standard deviation of the plausibility scores are as follows:

Mean	0.103
Standard deviation	0.037

---

<sup>8</sup>All figures are now quoted to three decimal places.

The following table shows plausibility scores for the learnt grammars using *Plausible*:

Sentence	$G2$	$G3$	$G4$
P1	0.100	0.075	0.100
P2	0.111	0.156	0.156
P3	0.111	0.156	0.156
P4	0.110	0.100	0.110
P5	0.083	0.083	0.083
P6	0.111	0.102	0.111
P7	0.054	0.048	0.054
P8	0.093	0.093	0.093
P9	0.071	0.071	0.071
P10	0.100	0.111	0.100
P11	0.104	0.104	0.104
P12	0.070	0.070	0.070
P13	0.095	0.048	0.095
P14	0.069	0.069	0.069
P15	0.103	0.082	0.103

The following table gives the arithmetic means and standard deviations of the plausibility scores:

	$G2$	$G3$	$G4$
Mean	0.092	0.091	0.098
Standard deviation	0.018	0.032	0.029

If we assume that the sentences used in the experiments are normally distributed, then we can determine how confident we are in the plausibility results. The null hypothesis is that there is no statistically significant difference between some pair of grammars. Given this hypothesis, the t-test results are as follows:

	$G_2$ and $G_4$	$G_3$ and $G_4$	$G_2$ and $G_3$
t-test	1.418	1.870	-1.300

The t-tests measure if the null hypothesis holds when comparing pairs of grammars. From these figures, we can be more than 90% confident that there is a significant

difference between grammars  $G_2$  and  $G_4$  and more than 95% confident that there is a difference between  $G_3$  and  $G_4$ . There is no evidence to suggest that there is a significant difference between  $G_2$  and  $G_3$ . Hence, we can conclude that  $G_4$  is more plausible than either  $G_2$  or  $G_3$ .  $G_4$  is less plausible than  $G_1$ . Interestingly, the combined learning process has acquired a grammar that is the best of either learning style in isolation. From the standard deviations (which measure how much a set of samples varies), the data-driven learner is the most consistently plausible grammar, whilst the manually constructed grammar is the least consistently plausible grammar.

### 6.3.4 Convergence results

Notionally, *Train* was partitioned into groups of 10 sentences and processed incrementally, using the learning configuration C. After dealing with each group, the resulting grammar was saved for subsequent inspection. This gave 6 grammars,  $G_{4_1} \dots G_{4_6}$ , where  $L(G_{4_i}) \subseteq L(G_{4_{i+1}})$  and  $G_{4_4} = G_4$ . The following table shows the growth in grammar size (measured in number of rules) with respect to training set size:

Grammar	Size
$G_1$	97
$G_{4_1}$	101
$G_{4_2}$	107
$G_{4_3}$	111
$G_{4_4}$	118
$G_{4_5}$	122
$G_{4_6}$	128

The graph 6.1 shows the system's learning curve in terms of the percentage of sentences parsed in *Test* with respect to training set size (in sentences). The graph 6.2 shows the increase in number of sentences generated in terms of the percentage of sentences parsed in *Test* and the size of the grammar.

As can be seen, convergence increases at a varying rate. Because of the small size of training material, nothing can be said about how much material is required

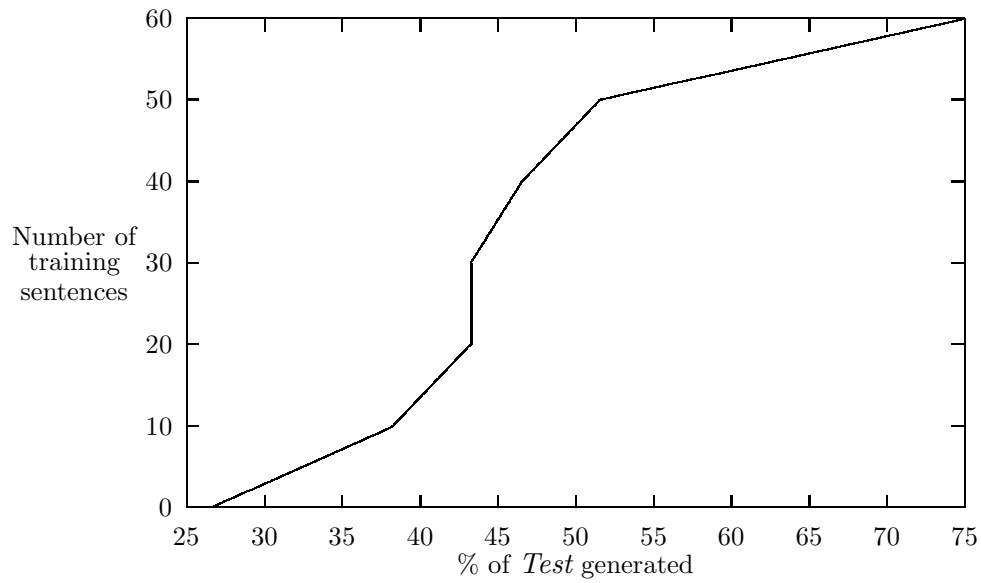


Figure 6.1: The system's learning curve

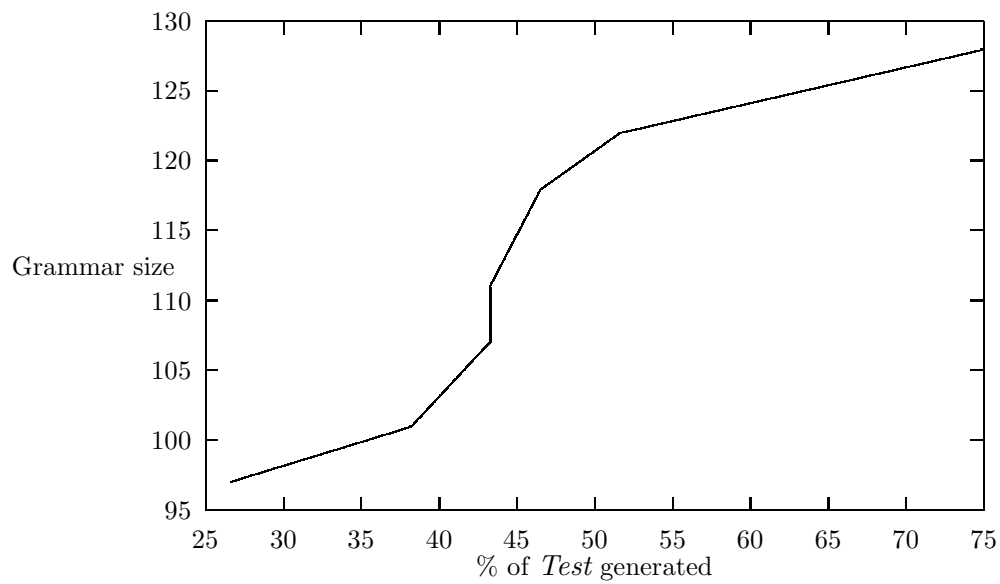


Figure 6.2: The increase in sentences generated

for the system to converge.<sup>9</sup> However, it is clear that only a moderate amount of

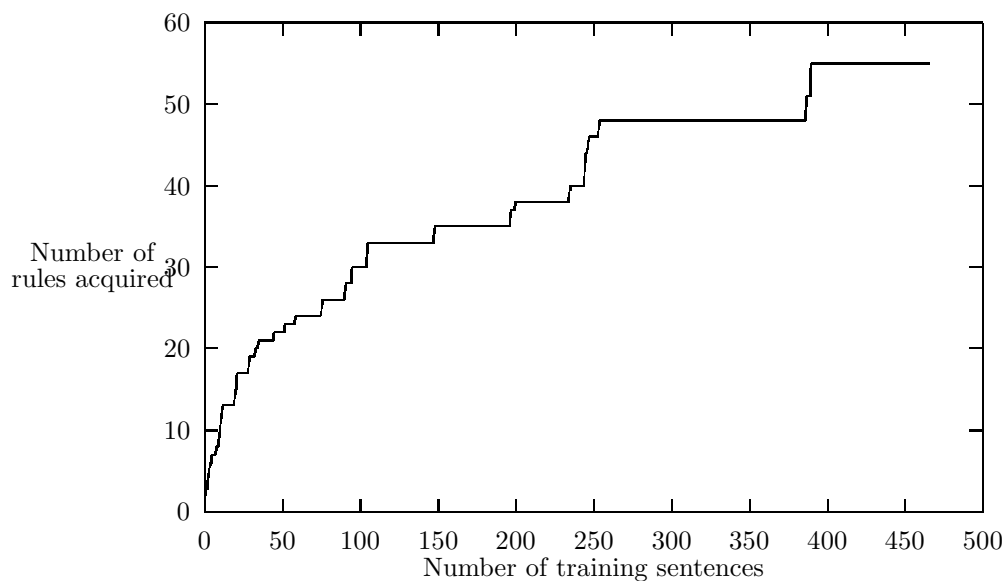


Figure 6.3: The learning curve when using more training sentences

material is necessary to allow a large increase in coverage.

This concludes the experiments. It is now time to draw some conclusions.

## 6.4 Discussion

The hypothesis, we tested by the experiments, was that combining both learning styles together would produce a quantitatively better grammar than would be produced when using either learning style in isolation.

From the plausibility results, we saw that *G4* was the most plausible, ranked equal first for the undergeneration test, and equal last for the overgeneration test. Hence, at least for plausibility, using both learning styles together produced a grammar that was quantitatively better than was produced when using either learning style in isolation. For the other tests, the outcome was less clear. *G4* certainly did

---

<sup>9</sup>Subsequent to the experiments reported in this thesis, the learner was configured to use model-based learning and was trained on 466 sentences taken from the SEC. This grammar parsed 98.3% of *Test*, 31% of *Bad* and attained a mean plausibility rating (against 48 sentences) of 0.102. The convergence garph is shown in figure 6.3. These results indicate that the results obtained from testing and training using small amounts of data are not that far from those obtained with larger testing and training sets.

well at dealing with undergeneration, but this was equalled by the performance of *G2*. *G4* also fared equally badly as *G2* for the overgeneration test. Both of these results show that, at least for the experiments reported here, the model, through incompleteness, made little impact upon the string sets generated by the grammars learnt. The results also show that data-driven learning has compensated for model-based incompleteness. That is, *G4* performance was equal to *G2*, despite the fact that *G4* also used a model as part of the learning process. Interestingly enough, there is a correlation between the size of the learnt grammars, in terms of disjunctive rules, and the degree of undergeneration and overgeneration: the larger the grammar, the larger the string set generated by that grammar. All the learnt grammars are larger than grammar *G1*, and indeed, they all undergenerate less than *G1*. From the convergence results, little can be said, other than that they hint that the system does not require much training material in order to allow a wide covering grammar to be learnt.

The conclusion from these experiments is that grammar learning does meet the success criteria for natural language grammars, but that this success is mainly due to data-driven learning. When model-based learning does have a role to play, it outperforms data-driven learning. Overall, using data-driven learning and model-based learning together is better than using data-driven learning or model-based learning in isolation.

## Chapter 7

# Conclusions

### 7.1 Introduction

This thesis described work on the machine learning of plausible unification-based grammars. In particular, the motivation of the work was to deal with the problem of undergeneration. Empirically, it was found that machine learning could deal with undergeneration, and in particular, that the combined use of data-driven learning and model-based learning produced more plausible grammars than when using either learning style in isolation. Other advances reported in this thesis include:

- a detailed discussion of what undergeneration is, why it occurs and how it can be successfully dealt with.
- a system that is capable of correctly treating undergeneration, is noise resistant, can learn about ambiguity, uses a parsimonious grammar formalism, is capable of being used in conjunction with a sentence corrector and allows experimentation between model-based and data-driven learning.
- an evaluation methodology for measuring grammar quality that is stricter than other approaches reported in the literature.

In order to achieve these results, numerous assumptions have been made. Section 7.2 discusses these and considers which assumptions can be weakened, and which must be held. Section 7.3 outlines ways in which grammar learning can be enhanced as a solution to the problem of dealing with undergeneration. Finally, section 7.4



ends the thesis with some general comments about the field of grammar learning.

## 7.2 Assumptions

The assumptions made in this thesis can be divided into two groups: those made for practical reasons, and those made for theoretical reasons. The former group have a conjectural status, in that they can be given-up or altered, whilst the latter group have an axiomatic status, in that they cannot be given up so easily.

Assumptions in the former group include:

- Having a complete lexicon.
- Using unary and binary rules.

The system presupposes that each word encountered was present within the lexicon. In practise, this meant using a stochastic tagger to assign a part-of-speech to any word that the system encountered. What is unsatisfactory about using a stochastic tagger is the coarse nature of the tag sets used. For example, the CLAWS2a tagset [6], which is used to label words in the Spoken English Corpus, lacks verbal subcategorisation and so grammars learnt will overgenerate more than they need to. Furthermore, lexical taggers tend to suffer degraded performance when tagging corpora other than the corpus used to train the tagger, thereby limiting the choice of language that can be successfully processed [80]. An avoidance of this coarseness problem and the lack of transportability would be to use a richer lexical representation (for example that used by the ACQUILEX Project [107]). This still runs the risk that the lexicon is incomplete. A method of ensuring that the lexicon was both complete, and also sufficiently rich, would be to learn the necessary entries. There has been some work in this area (for example Russell’s thesis [103]) which could be used to weaken the complete lexicon assumption.

There are a variety of ways of weakening the assumption that rules are either unary or binary. A first approach might be to fold rules that are frequently used together.<sup>1</sup> For example, assuming only using binary super rules, the system might learn rules for ditransitives such as  $VP \rightarrow V1\ NP$  and  $V1 \rightarrow V0\ NP$ . These could

---

<sup>1</sup>Samuelsson and Rayner’s use of EBL is similar to rule folding [106].

then be collapsed into the single rule  $VP \rightarrow V0\ NP\ NP$ . This has the advantage that the number of super rules are minimised (i. e. there is no need for a super rule with three RHS categories), which reduces the search space. However, it makes the rule construction less declarative. The number of categories in the learnt RHS's will no longer exactly relate to the number of categories in the instantiated super rule. A second approach might be to expand the super rules to include rules with more categories in their RHS. Learning would then proceed as before, except that this time, the learner might try to relate the length of rule RHSs with plausibility. This information would then be used to reduce the super rule set. However, expanding the super rules would increase the search space.

To learn about gapping, the system could use a 0-ary super rule. This would need to be refined or rejected by appropriate principles of grammaticality. As should be apparent, gap learning is a hard problem. One reason for this difficulty is that a sentence with gaps can be viewed as a longer version of that same sentence without gaps:

**40** *What did Tony have accepted today?*

**41** *What did Tony have accepted today e?*

Here, sentence 40 has a length of 6 words, whilst sentence 41 has a length of 7 words (including the gap). Since gaps can appear almost anywhere in the sentence, and from section 3.5.4, we saw that grammar learning is at least exponential with respect to sentence length, gap learning will introduce a dramatic expansion of the search space. Another reason for the difficulty in learning about unbounded dependencies is that gaps are hypothesised in relation to non-local, potentially erroneous information in the parse tree. This means that the system will wastefully construct gaps based upon structures that are later thrown away. The only way to deal with both of these sources of intractability involved with gap learning would be to use a strong model of grammaticality.

Assumptions in the latter group include:

- Using a formalism that is (at least) context free.
- Syntax is important for NLP.

- Learning competence, and not performance grammars.

The system does not learn nonterminal symbols, and hence cannot be used to learn the formal complexity of the language being learnt. So, the system cannot learn if the language is finite state, context free, or any of the other classes of language in the Chomsky hierarchy. Hence, it is necessary to fix the computational power of the grammar formalism prior to learning. Fixing the grammar formalism's power is a claim on the power of the language being learnt. In this work, natural languages are stated as being (at least) context free.

Another assumption is that syntax is important in its own right, and has its own role to play. Quite apart from contemporary theories of semantic interpretation (which require a distinct, separate grammar), other aspects of applied linguistics require grammars. For example, corpora can be parsed and used as linguistic databases for exploration. Robust grammar checkers, which are useful when teaching first and second languages to students, require a grammar. Grammars can therefore be assumed as being important.

The final assumption is the most problematic. As was stated in various parts of this thesis, the learner tries to learn competence and not performance grammars. As should be clear, making this distinction is controversial. The defence of learning a competence grammar is that a distinction can be made between ungrammatical and grammatical sentences. We contend that a grammar that fails to make this distinction becomes a vacuous theory and of little value. However, it is not clear if such a distinction can always be made, and hence, it could be said that the learner at times makes an arbitrary decision regarding the grammaticality/ungrammaticality distinction. Whilst this may be so, the converse, of allowing the learner to acquire performance grammars, is even less satisfactory. For example, a performance grammar trained using Scottish speakers would reflect Scottish performance. Such a grammar would need to be retrained if it were subsequently to be used in processing the language of (say) Welsh speakers. This would not be the case for a competence grammar. Hence, the task of acquisition is assumed to be one of learning competence, and not performance grammars.

### 7.3 Further work

There are numerous ways of extending this work. As was clear from the experiments in chapter six, the model of grammaticality had little impact upon the quality of the grammars. Hence, an obvious step would be to increase the contribution that the model can make. Areas of incompleteness could be identified by analysing the contributions that each of the principles of grammaticality had to make, and noting areas where they were deficient. Another step would be to use other principles of grammaticality, drawn from theories such as Government and Binding Theory [23]. For example, the learner does not deductively learn about long distance dependencies. Using GB principles such as subadjacency and Move- $\alpha$  would be a way of achieving this.

The learner could try to use textuality as a constraining device upon the quality of grammars. It is a known fact that the ability to predict which sentence is to come next increases with the number of prior sentences considered [110]. Hence, unexpected analyses could be detected and rejected, thereby preventing the learner from acquiring implausible grammars. As far as is known, considering textuality when learning grammars is completely novel. Related to the use of textuality would be using punctuation as another constraining device. Jones shows how punctuation can help reduce the syntactic ambiguity in long sentences [61], and so punctuation would help to identify constituents when learning.

Turning to data-driven learning, the language model would benefit from using a better approximation of the Noisy Channel Model. This could be achieved by using lexical co-occurrence statistics, by trying to reduce the entropy of the language generated by the learnt grammar, by using a more principled decision theory, and so on.

So far, the further work mentioned has only considered the two learning styles. It would also be possible to exploit the grammar formalism and try to generalise the rules. As it stands, the learner constructs rules that, being the result of unification, might at times be too specific. Interesting work would be to try to determine which of the features in the rules could be uninstantiated, and which of these features should be re-entrant.

The learner used a *rule*-based grammar formalism. An interesting possibility would be to consider using instead a *lexically*-based formalism. This would have the advantage of allowing syntactic, semantic, and pragmatic constraints to be expressed within the single framework [94].

Finally, it will be worthwhile to consider how the language learner relates to theories of human language acquisition. For example, one could view the data-driven component as corresponding to a set of parameters that need to be set, and the model-based component as corresponding to a set of principles. The experiments might therefore shed light on the relationship between these two aspects of language learning.

## 7.4 General conclusions

When this research started just under three years ago, not many grammar learning researchers were considering what linguistics had to offer. Now, this is no longer true, and the grammar learning community are tentatively beginning to use linguistic theories of universal grammar. One of the contributions of this thesis has been to show that a whole-hearted use of model-based learning can overcome the problems associated with data-driven grammar learning. Hence, data-driven approaches such as the Inside-Outside algorithm will benefit greatly from using model-based learning. Completeness can be combined with quality, thereby allowing wide covering grammars to be constructed that lend themselves to semantic interpretation. Hopefully, gone will be the days when researchers use just data-driven learning, coupled with inadequate formalisms, to try to acquire grammars that cannot be used for any task other than for language recognition purposes.

## Appendix A

### The Lexicon

The experiments reported in chapter seven used the SEC as a source of training and testing material. However, in common with other researchers, the Grammar Garden parsed SEC tag sequences, and not SEC sentences. This is for logistical reasons: it is far easier to create a lexicon of a few hundred tags than it is to create a lexicon of millions of words. So, a SEC entry:

```
[N It_PPH1 N]
[V 's_VBZ [N a_AT1 useful_JJ reminder_NN1
           [Fn that_CST
            [N some_DD scientists_NN2 N]
            [V find_VV0 [N Don_NP1 Cupitt_NP1 N]
            unscientific_JJ V]Fn]N]V] ._.
```

would be preprocessed into the tag sequence:

**42** *PPH1 VBZ AT1 JJ NN1 CST DD NN2 VV0 NP1 NP1 JJ*

for use by the Grammar Garden.

Grammar G1 contained the following lexicon:

$$\begin{aligned} \$ &\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, POSS +, NTYPE POSS, \\ WH -, CONJ - \end{array} \right] \\ APP\$ &\mapsto \left[ MINOR DET, POSS +, WH - \right] \\ AT &\mapsto \left[ MINOR DET, POSS -, WH - \right] \\ AT1 &\mapsto \left[ MINOR DET, PLU -, POSS -, WH - \right] \end{aligned}$$

BCS	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT SCOMP \right]$
BTO	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT VPINF \right]$
CC	$\mapsto$	$\left[ MINOR CONJ, CJTYPE END \right]$
CCB	$\mapsto$	$\left[ MINOR CONJ, CJTYPE END \right]$
CF	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT SFIN, CONJ - \right]$
CS	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT SFIN, CONJ - \right]$
CSA	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, PFORM AS, CONJ - \right]$
CSN	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, PFORM THAN, CONJ - \right]$
CST	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT SFIN, \right.$ $\left. PFORM THAT, CONJ - \right]$
CSW	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT SFIN, \right.$ $\left. PFORM WH, CONJ - \right]$
CSW	$\mapsto$	$\left[ N -, V -, BAR 0, MINOR NONE, SUBCAT VPINF, \right.$ $\left. PFORM WH, CONJ - \right]$
DA	$\mapsto$	$\left[ N +, V +, BAR 0, MINOR NONE, ATYPE ATT, \right.$ $\left. AFORM NONE, ADV -, CONJ - \right]$
DA	$\mapsto$	$\left[ N +, V -, BAR 0, MINOR NONE, POSS -, NTYPE PRO, \right.$ $\left. WH -, CONJ - \right]$
DA	$\mapsto$	$\left[ N +, V -, BAR 2, MINOR NONE, POSS -, NTYPE PRO, \right.$ $\left. WH -, CONJ - \right]$
DA	$\mapsto$	$\left[ MINOR DET, POSS -, WH - \right]$
DA1	$\mapsto$	$\left[ N +, V +, BAR 0, MINOR NONE, ATYPE ATT, \right.$ $\left. AFORM NONE, ADV -, CONJ - \right]$
DA1	$\mapsto$	$\left[ N +, V -, BAR 0, MINOR NONE, PLU -, POSS -, NTYPE PRO, \right.$ $\left. WH -, CONJ - \right]$
DA1	$\mapsto$	$\left[ N +, V -, BAR 2, MINOR NONE, PLU -, POSS -, NTYPE PRO, \right.$ $\left. WH -, CONJ - \right]$
DA1	$\mapsto$	$\left[ MINOR DET, PLU -, POSS -, \right.$ $\left. WH - \right]$
DA2	$\mapsto$	$\left[ N +, V +, BAR 0, MINOR NONE, ATYPE ATT, \right.$ $\left. AFORM NONE, ADV -, CONJ - \right]$
DA2	$\mapsto$	$\left[ N +, V -, BAR 0, MINOR NONE, PLU +, POSS -, NTYPE PRO, \right.$ $\left. WH -, CONJ - \right]$

DA2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DA2	$\mapsto \left[ MINOR DET, PLU +, POSS -, WH - \right]$
DA2R	$\mapsto \left[ N +, V +, BAR 0, MINOR NONE, AFORM ER, ADV -, CONJ - \right]$
DA2R	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, POSS -, \\ NTYPE PRO, WH -, CONJ - \end{array} \right]$
DA2R	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, \\ NTYPE PRO, WH -, CONJ - \end{array} \right]$
DA2R	$\mapsto \left[ MINOR DET, PLU +, POSS -, WH - \right]$
DAR	$\mapsto \left[ N +, V +, BAR 0, MINOR NONE, AFORM ER, ADV -, CONJ - \right]$
DAR	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DAR	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DAR	$\mapsto \left[ MINOR DET, PLU +, POSS -, WH - \right]$
DAT	$\mapsto \left[ N +, V +, BAR 0, MINOR NONE, AFORM EST, ADV -, CONJ - \right]$
DAT	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DAT	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DAT	$\mapsto \left[ MINOR DET, PLU +, POSS -, WH - \right]$
DB	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, POSS -, NTYPE PART, \\ WH -, CONJ - \end{array} \right]$
DB	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DB2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, POSS -, NTYPE PART, \\ WH -, CONJ - \end{array} \right]$
DB2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DD	$\mapsto \left[ MINOR DET, POSS -, WH - \right]$
DD	$\mapsto \left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, POSS -, NTYPE PRO, \\ WH -, CONJ - \end{array} \right]$
DD1	$\mapsto \left[ MINOR DET, PLU -, POSS -, WH - \right]$



DD1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, POSS -, NTYPE\ PRO, \\ WH -, CONJ - \end{array} \right]$
DD2	$\mapsto \left[ MINOR\ DET, PLU +, POSS -, WH - \right]$
DD2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, POSS -, NTYPE\ PRO, \\ WH -, CONJ - \end{array} \right]$
DDQ	$\mapsto \left[ MINOR\ DET, POSS -, WH + \right]$
DDQ\$	$\mapsto \left[ MINOR\ DET, POSS -, WH + \right]$
DDQV	$\mapsto \left[ MINOR\ DET, POSS -, WH + \right]$
EX	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, POSS -, NTYPE\ THERE, \\ WH -, CONJ - \end{array} \right]$
ICS	$\mapsto \left[ N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ SFIN \right]$
ICS	$\mapsto \left[ N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ VPING, CONJ - \right]$
IF	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 0, MINOR\ NONE, \\ SUBCAT\ SINF, PFORM\ FOR, CONJ - \end{array} \right]$
IF	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 0, MINOR\ NONE, \\ SUBCAT\ NP, PFORM\ FOR, CONJ - \end{array} \right]$
II	$\mapsto \left[ N -, V -, BAR\ 0, MINOR\ NONE, CONJ - \right]$
IO	$\mapsto \left[ N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ NP, PFORM\ OF, CONJ - \right]$
IO	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ VPING, \\ PFORM\ OF, CONJ - \end{array} \right]$
IW	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ NP, \\ PFORM\ WITH, CONJ - \end{array} \right]$
IW	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 0, MINOR\ NONE, SUBCAT\ VPING, \\ PFORM\ WITHOUT, CONJ - \end{array} \right]$
JA	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ PRD, AFORM\ NONE, \\ ADV -, CONJ - \end{array} \right]$
JA	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ PRD, AFORM\ NONE, \\ ADV -, CONJ - \end{array} \right]$
JB	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ ATT, AFORM\ NONE, \\ ADV -, CONJ - \end{array} \right]$
JB	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ ATT, AFORM\ NONE, \\ ADV -, CONJ - \end{array} \right]$
JBR	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ ATT, AFORM\ ER, \\ ADV -, CONJ - \end{array} \right]$

JBT	$\mapsto \begin{bmatrix} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ ATT, AFORM\ EST, \\ ADV -, CONJ - \end{bmatrix}$
JJ	$\mapsto \begin{bmatrix} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ NONE, ADV -, CONJ - \end{bmatrix}$
JJ	$\mapsto \begin{bmatrix} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ NONE, ADV -, CONJ - \end{bmatrix}$
JJR	$\mapsto \begin{bmatrix} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ ER, ADV -, CONJ - \end{bmatrix}$
JJR	$\mapsto \begin{bmatrix} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ ER, ADV -, CONJ - \end{bmatrix}$
JJT	$\mapsto \begin{bmatrix} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ EST, ADV -, CONJ - \end{bmatrix}$
JJT	$\mapsto \begin{bmatrix} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ EST, ADV -, CONJ - \end{bmatrix}$
JK	$\mapsto \begin{bmatrix} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ CAT, \\ AFORM\ NONE, ADV -, CONJ - \end{bmatrix}$
LE	$\mapsto \begin{bmatrix} MINOR\ CONJ, CJTYPE\ BEGIN \end{bmatrix}$
MC	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, POSS -, NTYPE\ NUM, WH -, CONJ - \end{bmatrix}$
MC	$\mapsto \begin{bmatrix} MINOR\ DET, PLU +, POSS -, WH - \end{bmatrix}$
MC\$	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, POSS +, NTYPE\ NUM, WH -, CONJ - \end{bmatrix}$
MC\$	$\mapsto \begin{bmatrix} MINOR\ DET, PLU +, POSS +, \\ WH - \end{bmatrix}$
MC-MC	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, POSS -, NTYPE\ NUM, \\ WH -, CONJ - \end{bmatrix}$
MC-MC	$\mapsto \begin{bmatrix} MINOR\ DET, PLU +, POSS +, WH - \end{bmatrix}$
MC1	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, POSS -, \\ NTYPE\ NUM, WH -, CONJ - \end{bmatrix}$
MC1	$\mapsto \begin{bmatrix} MINOR\ DET, PLU -, POSS +, WH - \end{bmatrix}$
MC2	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, POSS -, \\ NTYPE\ NUM, WH -, CONJ - \end{bmatrix}$
MC2	$\mapsto \begin{bmatrix} MINOR\ DET, PLU +, POSS +, WH - \end{bmatrix}$
MD	$\mapsto \begin{bmatrix} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ NUM, AFORM\ NONE, \\ ADV -, CONJ - \end{bmatrix}$
MD	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, POSS -, NTYPE\ NUM, \\ WH -, CONJ - \end{bmatrix}$
MF	$\mapsto \begin{bmatrix} N +, V -, BAR\ 2, MINOR\ NONE, POSS -, NTYPE\ NUM, \\ WH -, CONJ - \end{bmatrix}$
NC2	$\mapsto \begin{bmatrix} N +, V -, BAR\ 0, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{bmatrix}$

ND1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE DIR, WH -, CONJ - \end{array} \right]$
ND1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU -, \\ POSS -, NTYPE DIR, WH -, CONJ - \end{array} \right]$
NN	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, \\ POSS -, NTYPE NORM, WH -, \\ CONJ - \end{array} \right]$
NN	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, \\ POSS -, NTYPE NORM, WH -, \\ CONJ - \end{array} \right]$
NN1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE NORM, \\ WH -, CONJ - \end{array} \right]$
NN1\$	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS +, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NN2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NN2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNJ	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, \\ POSS -, NTYPE NORM, \\ WH -, CONJ - \end{array} \right]$
NNJ1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNJ2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNJ2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNL	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNL1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$
NNL2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE NORM, WH -, CONJ - \end{array} \right]$

NNL2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NNO	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NNO1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NNO2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NN02	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NNS	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, \\ POSS -, NTYPE\ NORM, WH -, CONJ - \end{array} \right]$
NNS1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ TIT, WH -, CONJ - \end{array} \right]$
NNS2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ TIT, WH -, CONJ - \end{array} \right]$
NNS2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ TIT, WH -, CONJ - \end{array} \right]$
NNSA1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ POSTTIT, WH -, CONJ - \end{array} \right]$
NNSA2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ POSTTIT, WH -, CONJ - \end{array} \right]$
NNSB	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, \\ POSS -, NTYPE\ PRETIT, WH -, CONJ - \end{array} \right]$
NNSB1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRETIT, WH -, CONJ - \end{array} \right]$
NNSB2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ PRETIT, WH -, CONJ - \end{array} \right]$
NNSB2	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ PRETIT, WH -, CONJ - \end{array} \right]$
NNT	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, \\ POSS -, NTYPE\ TEMP, WH -, CONJ - \end{array} \right]$
NNT1	$\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR\ 0, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ TEMP, WH -, CONJ - \end{array} \right]$

NNT2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NNT2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NNU $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NNU1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NNU2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NNU2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NP $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, \\ POSS -, NTYPE NAME, WH -, CONJ - \end{array} \right]$
NP1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE NAME, WH -, CONJ - \end{array} \right]$
NP1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU -, \\ POSS -, NTYPE NAME, WH -, CONJ - \end{array} \right]$
NP2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE NAME, WH -, CONJ - \end{array} \right]$
NPD1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NPD1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU -, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NPD2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NPD2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU +, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NPM1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU -, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$
NPM1 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 2, MINOR NONE, PLU -, \\ POSS -, NTYPE MEAS, WH -, CONJ - \end{array} \right]$
NPM2 $\mapsto$	$\left[ \begin{array}{l} N +, V -, BAR 0, MINOR NONE, PLU +, \\ POSS -, NTYPE TEMP, WH -, CONJ - \end{array} \right]$

NPM2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ MEAS, WH -, CONJ - \end{array} \right]$
PN	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PN1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PNQO	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH +, CONJ - \end{array} \right]$
PNQS	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH +, CONJ - \end{array} \right]$
PNQV\$	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH +, CONJ - \end{array} \right]$
PNQVO	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH +, CONJ - \end{array} \right]$
PNQVS	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH +, CONJ - \end{array} \right]$
PNX1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PP\$	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS +, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPH1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPHO1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPHO2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPHS1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, \\ NTYPE\ PRONOM, WH -, CONJ - \end{array} \right]$
PPHS2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, \\ NTYPE\ PRONOM, WH -, CONJ - \end{array} \right]$
PPIO1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$

PPIO2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPIS1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRONOM, WH -, CONJ - \end{array} \right]$
PPIS2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, \\ NTYPE\ PRONOM, WH -, CONJ - \end{array} \right]$
PPX1	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPX2	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU +, \\ POSS -, NTYPE\ PRO, WH -, CONJ - \end{array} \right]$
PPY	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ PRONOM, WH -, CONJ - \end{array} \right]$
RA	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ POST, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
REX	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ XCOMP, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RG	$\mapsto \left[ MINOR\ DEG \right]$
RGA	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ POST, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RGQ	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ HOW, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RGQV	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, ATYPE\ HOW, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RGR	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ ER, \\ ADV +, CONJ - \end{array} \right]$
RGR	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ ER, \\ ADV +, CONJ - \end{array} \right]$
RGT	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ EST, \\ ADV +, CONJ - \end{array} \right]$
RGT	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ EST, \\ ADV +, CONJ - \end{array} \right]$
RL	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RP	$\mapsto \left[ MINOR\ PRT \right]$

RPK	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, ATYPE\ CAT, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RR	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RR	$\mapsto \left[ \begin{array}{l} N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ NONE, \\ ADV +, CONJ - \end{array} \right]$
RRQ	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 1, MINOR\ NONE, SUBCAT\ NONE, \\ PFORM\ WH, CONJ - \end{array} \right]$
RRQV	$\mapsto \left[ \begin{array}{l} N -, V -, BAR\ 1, MINOR\ NONE, SUBCAT\ NONE, \\ PFORM\ WH, CONJ - \end{array} \right]$
RRR	$\mapsto \left[ N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ ER, ADV +, CONJ - \right]$
RRR	$\mapsto \left[ N +, V +, BAR\ 1, MINOR\ NONE, AFORM\ ER, ADV +, CONJ - \right]$
RRT	$\mapsto \left[ N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ EST, ADV +, CONJ - \right]$
RRT	$\mapsto \left[ N +, V +, BAR\ 0, MINOR\ NONE, AFORM\ EST, ADV +, CONJ - \right]$
RT	$\mapsto \left[ \begin{array}{l} N +, V -, BAR\ 2, MINOR\ NONE, PLU -, \\ POSS -, NTYPE\ TEMP, WH -, CONJ - \end{array} \right]$
TO	$\mapsto \left[ N -, V +, BAR\ 0, MINOR\ NONE, AUX\ TO, VFORM\ INF, CONJ - \right]$
UH	$\mapsto \left[ MINOR\ INTERJ \right]$
VB0	$\mapsto \left[ N -, V +, BAR\ 0, MINOR\ NONE, AUX\ BE, VFORM\ BSE, CONJ - \right]$
VBDR	$\mapsto \left[ N -, V +, BAR\ 0, MINOR\ NONE, AUX\ BE, VFORM\ PAST, CONJ - \right]$
VBDZ	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ PLU -, AUX\ BE, VFORM\ PAST, CONJ - \end{array} \right]$
VBG	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ AUX\ BE, VFORM\ ING, CONJ - \end{array} \right]$
VBM	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ PLU -, AUX\ BE, VFORM\ PRES, CONJ - \end{array} \right]$
VCN	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ AUX\ BE, VFORM\ PPART, CONJ - \end{array} \right]$
VBR	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ PLU +, AUX\ BE, VFORM\ PRES, CONJ - \end{array} \right]$
VBZ	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ PLU -, AUX\ BE, VFORM\ PRES, CONJ - \end{array} \right]$
VD0	$\mapsto \left[ \begin{array}{l} N -, V +, BAR\ 0, MINOR\ NONE, \\ AUX\ DO, VFORM\ BSE, CONJ - \end{array} \right]$



VDD	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX DO, VFORM PAST, CONJ - \end{array} \right]$
VDG	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX DO, VFORM ING, CONJ - \end{array} \right]$
VDN	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX DO, VFORM PPART, CONJ - \end{array} \right]$
VDZ	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ PLU -, AUX DO, VFORM PRES, CONJ - \end{array} \right]$
VH0	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX HAVE, VFORM BSE, CONJ - \end{array} \right]$
VHD	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX HAVE, VFORM PAST, CONJ - \end{array} \right]$
VHG	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX HAVE, VFORM ING, CONJ - \end{array} \right]$
VHN	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX HAVE, VFORM PPART, CONJ - \end{array} \right]$
VHZ	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ PLU -, AUX HAVE, VFORM PRES, CONJ - \end{array} \right]$
VM	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX MODAL, CONJ - \end{array} \right]$
VMK	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ AUX CAT, CONJ - \end{array} \right]$
VV0	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM BSE, CONJ - \end{array} \right]$
VVD	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM PAST, CONJ - \end{array} \right]$
VVG	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM ING, CONJ - \end{array} \right]$
VVGK	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM ING, CONJ - \end{array} \right]$
VVN	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM PPART, CONJ - \end{array} \right]$
VVNK	$\mapsto$	$\left[ \begin{array}{l} N -, V +, BAR 0, MINOR NONE, \\ VFORM PPART, CONJ - \end{array} \right]$

$$\begin{aligned}
\text{VVZ} &\mapsto \left[ \begin{array}{l} N -, V +, \text{BAR } 0, \text{MINOR NONE}, \\ \text{PLU } -, \text{VFORM PRES}, \text{CONJ } - \end{array} \right] \\
\text{XX} &\mapsto \left[ \text{MINOR NOT} \right] \\
\text{ZZ1} &\mapsto \left[ \begin{array}{l} N +, V -, \text{BAR } 0, \text{MINOR NONE}, \\ \text{PLU } -, \\ \text{POSS } -, \text{NTYPE NAME}, \text{WH } -, \text{CONJ } - \end{array} \right] \\
\text{ZZ1} &\mapsto \left[ \begin{array}{l} N +, V -, \text{BAR } 2, \text{MINOR NONE}, \text{PLU } -, \\ \text{POSS } -, \text{NTYPE NAME}, \text{WH } -, \text{CONJ } - \end{array} \right] \\
\text{ZZ2} &\mapsto \left[ \begin{array}{l} N +, V -, \text{BAR } 0, \text{MINOR NONE}, \text{PLU } +, \\ \text{POSS } -, \text{NTYPE NAME}, \text{WH } -, \text{CONJ } - \end{array} \right] \\
\text{ZZ2} &\mapsto \left[ \begin{array}{l} N +, V -, \text{BAR } 2, \text{MINOR NONE}, \text{PLU } +, \\ \text{POSS } -, \text{NTYPE NAME}, \text{WH } -, \text{CONJ } - \end{array} \right]
\end{aligned}$$

## Appendix B

# The Original Grammar

Here we present, in a paraphrased form, grammar *G1*. The symbols used are notational abbreviations for feature-structures and are of the form *Xn*, where *X* is a phrase, and *n* is the bar level. Minor categories are represented by idiosyncratic symbols. The correspondences between symbols and feature-structures are as follows:

Symbol	Feature-structure
N <sub>n</sub>	[ <i>N +, V -, BAR n, MINOR NONE</i> ]
V <sub>n</sub>	[ <i>N -, V +, BAR n, MINOR NONE</i> ]
A <sub>n</sub>	[ <i>N +, V +, BAR n, MINOR NONE</i> ]
P <sub>n</sub>	[ <i>N -, V -, BAR n, MINOR NONE</i> ]
INTERJ	[ <i>MINOR INTERJ</i> ]
DEG	[ <i>MINOR DEG</i> ]
NOT	[ <i>MINOR NOT</i> ]
CONJ	[ <i>MINOR CONJ</i> ]
DT	[ <i>MINOR DET</i> ]

Note that the grammars are paraphrased: the grammars in feature-structure form contain categories with 18 features. This means that some of the paraphrased rules contain apparent redundancies (such as  $\{N1, N1\} \rightarrow N1\ N1$ ). In reality, such rules, in full, do have distinct categories, but these distinctions are not preserved when paraphrased. If interested, the reader should contact the author (miles@minster.york.ac.uk) for full details.

Grammar  $G1$  is as follows:

$$A1 \rightarrow A1 \ A1$$

$$A1 \rightarrow DEG \ A1$$

$$A0 \rightarrow A0 \ A0$$

$$A1 \rightarrow A0 \ V1$$

$$A1 \rightarrow A0 \ V2$$

$$A1 \rightarrow A0 \ P1$$

$$A1 \rightarrow A0$$

$$P0 \rightarrow P0 \ P0$$

$$P2 \rightarrow A2 \ P1$$

$$P1 \rightarrow P0$$

$$P1 \rightarrow P0 \ V1$$

$$P1 \rightarrow P0 \ V1$$

$$P1 \rightarrow P0 \ V2$$

$$P1 \rightarrow P0 \ V2$$

$$P1 \rightarrow P0 \ N2$$

$$N1 \rightarrow A2 \ N1$$

$$N1 \rightarrow A1 \ N1$$

$$N1 \rightarrow N1 \ N1$$

$$N1 \rightarrow N1 \ P1$$

$$N0 \rightarrow N0 \ N0$$

$$N0 \rightarrow N0 \ N0$$

$$N1 \rightarrow N0 \ P1$$

$$N1 \rightarrow N0 \ V1$$

$$N1 \rightarrow N0 \ V2$$

$$N1 \rightarrow N0$$

$$N2 \rightarrow N2 \ N1$$

$$N2 \rightarrow DT \ N1$$

$$N2 \rightarrow DT \ N1$$

$$N2 \rightarrow N1$$

$$V1 \rightarrow NOT \ V1$$

$$V1 \rightarrow V0 \ V1$$

$V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ N2$   
 $V1 \rightarrow V0 \ A2$   
 $V1 \rightarrow V0 \ A1$   
 $V1 \rightarrow V0 \ P1$   
 $V1 \rightarrow V0 \ P2$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V1 \ V1$   
 $V1 \rightarrow V1 \ N2$   
 $V1 \rightarrow V1 \ P2$   
 $V1 \rightarrow V1 \ A2$   
 $V1 \rightarrow V1 \ A1$   
 $V0 \rightarrow V0 \ V0$   
 $V1 \rightarrow V0 \ N2 \ N2 \ V2$   
 $V1 \rightarrow V0 \ N2 \ A2$   
 $V1 \rightarrow V0 \ N2 \ A1$   
 $V1 \rightarrow V0 \ N2 \ V2$   
 $V1 \rightarrow V0 \ N2 \ P1$   
 $V1 \rightarrow V0 \ N2 \ V1$   
 $V1 \rightarrow V0 \ N2 \ V1$   
 $V1 \rightarrow V0 \ N2 \ N2$   
 $V1 \rightarrow V0 \ A2$   
 $V1 \rightarrow V0 \ A1$   
 $V1 \rightarrow V0 \ V2$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ V1$   
 $V1 \rightarrow V0 \ P1$   
 $V1 \rightarrow V0 \ N2$   
 $V0 \rightarrow V0 \ NOT$

$$A0 \rightarrow CONJ \quad A0$$

$$A1 \rightarrow CONJ \quad A1$$

$$P0 \rightarrow CONJ \quad P0$$

$$P1 \rightarrow CONJ \quad P1$$

$$N0 \rightarrow CONJ \quad N0$$

$$N1 \rightarrow CONJ \quad N1$$

$$N2 \rightarrow CONJ \quad N2$$

$$V0 \rightarrow CONJ \quad V0$$

$$V1 \rightarrow CONJ \quad V1$$

$$V2 \rightarrow CONJ \quad V2$$

$$A1 \rightarrow CONJ \quad A1$$

$$P1 \rightarrow CONJ \quad P1$$

$$N2 \rightarrow CONJ \quad N2$$

$$V1 \rightarrow CONJ \quad V1$$

$$V2 \rightarrow CONJ \quad V2$$

$$V2 \rightarrow COMP \quad V2$$

$$A2 \rightarrow A1 \quad A1$$

$$P2 \rightarrow A1 \quad P1$$

$$V1 \rightarrow A1 \quad V1$$

$$V2 \rightarrow A1 \quad V2$$

$$V2 \rightarrow N2 \quad V2$$

$$P1 \rightarrow P1 \quad P1$$

$$V2 \rightarrow P1 \quad V2$$

$$V2 \rightarrow P1 \quad V2$$

$$V2 \rightarrow A1 \quad V2$$

$$V2 \rightarrow N2 \quad V2$$

$$V2 \rightarrow N2 \quad V2$$

$$V2 \rightarrow V0 \quad V2$$

$$N2 \rightarrow N2 \quad N2$$

$$N2 \rightarrow N2 \quad N0$$

$$N2 \rightarrow N2 \quad A1$$

$$V2 \rightarrow N2 \quad V1$$

$$V2 \rightarrow V2 \ V2$$

$$S3 \rightarrow V2$$

## Appendix C

# The Learnt Grammars

Here we present the grammars learnt in chapter seven. These listings use the same paraphrasing as in appendix B and only show the rules learnt. They do not repeat those rules in grammar  $G1$ .

Grammar  $G2$  is:

$$\begin{aligned} \{V1, N3, N2, V0\} &\rightarrow N2 \ V0 \\ \{N2, N1, N0, N1\} &\rightarrow N0 \ N1 \\ \{N2, P1, P0, N3 \ N1, N0\} &\rightarrow P0 \ \{N0, N1, N3\} \\ \{P2, S3, P1\} &\rightarrow S3 \ P1 \\ \{V1, P2, P1, V0\} &\rightarrow P1 \ V0 \\ \{N1, N3, N2, N0\} &\rightarrow N2 \ N0 \\ \{V1, V1, V0, V0\} &\rightarrow V0 \ V0 \\ \{V2, V2, N2, N1, V1, V1, V0, V0\} &\rightarrow N1 \ \{V0, V0, V1, V1\} \\ \{N2, N1, N0, N1\} &\rightarrow N0 \ N1 \\ \{N2, V1, V0, N1\} &\rightarrow V0 \ N1 \\ \{V1, N3, N2, V0\} &\rightarrow N2 \ V0 \\ \{N2, V1, N3, N2, V0, N1\} &\rightarrow N2 \ \{N1, V0\} \\ \{V2, V1, N3, N2, V0\} &\rightarrow INTERJ \ \{V0, N2, N3, V1\} \\ \{V1, V0\} &\rightarrow INTERJ \ V0 \\ \{A2, A2, A1, A1\} &\rightarrow A1 \ A1 \\ \{V1, V1, V0, V0\} &\rightarrow V0 \ V0 \\ \{V1, V1, N3, N2, V0, V0\} &\rightarrow N2 \ \{V0, V0\} \end{aligned}$$



$$\begin{aligned}
& \{A1, V1, V0, A0\} \rightarrow V0 \quad A0 \\
& \{V1, N2, N1, V0\} \rightarrow N1 \quad V0 \\
& \{N3, N1, N0, N2\} \rightarrow N0 \quad N2 \\
& \{P1, N1, N0, P0, N3\} \rightarrow N0 \quad \{N3, P0\} \\
& \{V2, A2, N1, N0, A1, V1, V0, A0\} \rightarrow N0 \quad \{A0, V0, V1, A1\} \\
& \{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
& \{N2, N2, N3, N2, N1, N1\} \rightarrow N2 \quad \{N1, N1\} \\
& \{N2, N3, N2, N1\} \rightarrow N2 \quad N1 \\
& \{V1, N3, N2, V0\} \rightarrow N2 \quad V0 \\
& \{V1, V2, V1, V0\} \rightarrow V1 \quad V0 \\
& \{S3, V2, V1, V2, V1, V0\} \rightarrow DT \quad \{V0, V1, V2, V1\} \\
& \{V1, V0\} \rightarrow DT \quad V0 \\
& \{N3, N1, N0, N2\} \rightarrow N0 \quad N2 \\
& \{N1, N3, N0, N2\} \rightarrow DT \quad \{N2, N0, N3\} \\
& \{N3, N2, A1, A0, N1, N3, N0, N2\} \rightarrow A0 \quad \{N2, N0, N3, N1\}
\end{aligned}$$

Grammar  $G3$  is:

$$\begin{aligned}
& \{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
& \{N2, P1, P0, N3, N1, N0\} \rightarrow P0 \quad \{N0, N1, N3\} \\
& \{A2, V1, V0, A1\} \rightarrow V0 \quad A1 \\
& \{P2, S3, P1\} \rightarrow S3 \quad P1 \\
& \{V1, P2, P1, V0\} \rightarrow P1 \quad V0 \\
& \{N1, N3, N2, N0\} \rightarrow N2 \quad N0 \\
& \{V1, V1, V0, V0\} \rightarrow V0 \quad V0 \\
& \{V2, V2, N2, N1, V1, V1, V0, V0\} \rightarrow N1 \quad \{V0, V0, V1, V1\} \\
& \{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
& \{N2, V1, V0, N1\} \rightarrow V0 \quad N1 \\
& \{V1, N3, N2, V0\} \rightarrow N2 \quad V0 \\
& \{N2, V1, N3, N2, V0, N1\} \rightarrow N2 \quad \{N1, V0\} \\
& \{V2, V1, N3, N2, V0\} \rightarrow INTERJ \quad \{V0, N2, N3, V1\} \\
& \{V1, V0\} \rightarrow INTERJ \quad V0 \\
& \{A2, A2, A1, A1\} \rightarrow A1 \quad A1
\end{aligned}$$

$$\begin{aligned}
&\{V1, V1, V0, V0\} \rightarrow V0 \quad V0 \\
&\{V1, V1, N3, N2, V0, V0\} \rightarrow N2 \quad \{V0, V0\} \\
&\{V1, N2, N1, V0\} \rightarrow N1 \quad V0 \\
&\{N3, N1, N0, N2\} \rightarrow N0 \quad N2 \\
&\{N1, N0, N3\} \rightarrow N0 \quad N3 \\
&\{V2, N1, N0, V1\} \rightarrow N0 \quad V1 \\
&\{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
&\{N2, N2, N3, N2, N1, N1\} \rightarrow N2 \quad \{N1, N1\} \\
&\{N2, N3, N2, N1\} \rightarrow N2 \quad N1 \\
&\{V1, A3, N3, N2, A2, V0\} \rightarrow N2 \quad \{V0, A2\} \\
&\{V1, N3, N2, V0\} \rightarrow N2 \quad V0 \\
&\{V1, V2, V1, V0\} \rightarrow V1 \quad V0 \\
&\{S3, V2, V1, V2, V1, V0\} \rightarrow DT \quad \{V0, V1, V2, V1\} \\
&\{V1, V0\} \rightarrow DT \quad V0 \\
&\{N3, N3, N2, N2\} \rightarrow N2 \quad N2 \\
&\{A1, A0, N3, N3, N2, N2\} \rightarrow A0 \quad \{N2, N2, N3, N3\}
\end{aligned}$$

Finally, grammar  $G4$  is:

$$\begin{aligned}
&\{V1, N3, N2, V0\} \rightarrow N2 \quad V0 \\
&\{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
&\{N2, P1, P0, N3, N1, N0\} \rightarrow P0 \quad \{N0, N1, N3\} \\
&\{P2, S3, P1\} \rightarrow S3 \quad P1 \\
&\{V1, P2, P1, V0\} \rightarrow P1 \quad V0 \\
&\{N1, N3, N2, N0\} \rightarrow N2 \quad N0 \\
&\{V1, V1, V0, V0\} \rightarrow V0 \quad V0 \\
&\{V2, V2, N2, N1, V1, V1, V0, V0\} \rightarrow N1 \quad \{V0, V0, V1, V1\} \\
&\{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
&\{N2, V1, V0, N1\} \rightarrow V0 \quad N1 \\
&\{V1, N3, N2, V0\} \rightarrow N2 \quad \{V0\} \\
&\{N2, V1, N3, N2, V0, N1\} \rightarrow N2 \quad \{N1, V0\} \\
&\{V2, V1, N3, N2, V0\} \rightarrow INTERJ \quad \{V0, N2, N3, V1\} \\
&\{V1, V0\} \rightarrow INTERJ \quad V0
\end{aligned}$$

$$\begin{aligned}
& \{A2, A2, A1, A1\} \rightarrow A1 \quad A1 \\
& \{V1, V1, V0, V0\} \rightarrow V0 \quad V0 \\
& \{V1, V1, N3, N2, V0, V0\} \rightarrow N2 \quad \{V0, V0\} \\
& \{A1, V1, V0, A0\} \rightarrow V0 \quad A0 \\
& \{V1, N2, N1, V0\} \rightarrow N1 \quad V0 \\
& \{N3, N1, N0, N2\} \rightarrow N0 \quad N2 \\
& \{N1, N0, N3\} \rightarrow N0 \quad N3 \\
& \{V2, A2, N1, N0, A1, V1, V0, A0\} \rightarrow N0 \quad \{A0, V0, V1, A1\} \\
& \{N2, N1, N0, N1\} \rightarrow N0 \quad N1 \\
& \{N2, N2, N3, N2, N1, N1\} \rightarrow N2 \quad \{N1, N1\} \\
& \{N2, N3, N2, N1\} \rightarrow N2 \quad N1 \\
& \{V1, N3, N2, V0\} \rightarrow N2 \quad V0 \\
& \{V1, V2, V1, V0\} \rightarrow V1 \quad V0 \\
& \{S3, V2, V1, V2, V1, V0\} \rightarrow DT \quad \{V0, V1, V2, V1\} \\
& \{V1, V0\} \rightarrow DT \quad \{V0\} \\
& \{N3, N1, N0, N2\} \rightarrow N0 \quad N2 \\
& \{N1, N3, N0, N2\} \rightarrow DT0 \quad \{N2, N0, N3\} \\
& \{N3, N2, A1, A0, N1, N3, N0, N2\} \rightarrow A0 \quad \{N2, N0, N3, N1\}
\end{aligned}$$

## Appendix D

# The Corpus Material Used in Evaluation

In this appendix we present the tag sequences used to train the grammars (*Train*), the tag sequences used to evaluate overgeneration (*Bad*), the tag sequences used to test for undergeneration (*Test*), and the parses for both *Yardstick* and *Plausible*. For brevity, the actual SEC sentences corresponding to the SEC tag sequences have been omitted. The interested reader should contact the author (miles@minster.york.uk.ac) for details of these sentences.

*Train* consisted of the tag sequences:

RR AT NN2 RT

II AT JJ NNJ

MC1 NN1 VBZ JJ

DD1 VBZ NP1 NP1

DD1 VBZ NP1 NP1

DDQ VDD PPY VV0

PNQS VDD PPY VV0

PPIS1 VH0 AT1 NN1

AT NN1 VVZ RP

AT JJ NN1

RT NN1 NN1

NNJ NN1 NN1

NP1 NP1 VVZ

NNJ NN1 NN1

NN1 VVD JJ

CC APP\$ NN1

PPHS2 VBDR VVN

RGQ RR RL

UH VV0 RP

PPY VH0 NN1

UH PPIS1 VVD

PPIS2 VVD RP

RR PPHS1 VVD

DA1 NN1 RL

NNSB1 NP1

NNSB1 NP1

AT NN1

PPHS1 VVD

JJ NN1

RR RR

AT NN1

NP1>NNL2

JJ NN1

PN1 RA

NP1 NP1

UH UH

VDD PPHS1

VV0 RP

NP1 VVD

PPIS1 VV0

RRQ XX

CC NN1

AT NN1

PPIS1 VM

VV0 PPY  
 CC PPX1  
 JJ NN1  
 UH VDN  
 PPH1 VBZ  
 PPHS1 VVD  
 NN1 NN2  
 VV0 RL  
 VM PPY  
 PPHS1 VVD  
 AT1 NNT1  
 NP1 NP1  
 NNSB1 NP1  
 AT NN1  
 PPHS1 VVD

*Bad* consisted of the tag sequences:

LE UH MC1 VHN RRT DA1  
 JK MD NNT2 DA2 ZZ2>NNL1  
 CST CSA RRR MC-MC>NNL2 RGQV  
 NNU1 VVN MC-MC AT1 VVNK JJ  
 DDQ NNU2 CCB VDG RGT NNT1  
 NNJ2 VBN DAR CC PNQV\$ VB0  
 RGR ZZ2 RRR CSA DB RGQV  
 CSA PNQVO>NNJ2 DAT XX VBDR  
 UH PNQS PNQVS PPY DAR DA2R  
 IW>NNJ1>NNL2 TO JJR NPD1  
 CCB>NN2>NNS1>NNO1>NNS2 DD1  
 RGA NNSB1 VHD PNQS JBR \$  
 IF VM PNQS NP2 VDN RRQ  
 PPHS2 DDQV VB0 VDN NNU1>VVZ  
 ICS VHN NP1 MD JJ RGA  
 AT NNSB2 MC-MC>NNO1>VBN MC1

VDN NNS DDQ IF \$ JJT  
PPX2>NNL1 \$>NNL UH NPD1  
JA BTO VD0 VBN VHN DDQ\$  
RRQV PNQVO NP2 PN1 NPD2 MF  
VH0 DA2 REX NNT1 NNS>NNJ2  
NC2 XX VB0 PNQV\$ PNQV\$ MC  
NNL VDN>NN1\$>NNS1 PNQV\$ DDQ\$  
II VBZ PPX1>NNJ1 VMK LE  
MD>NNL DDQV MF VBN PPHO1  
DA1 PN II MC\$ VBM MC-MC  
CSA>NNS1 RL VDD>JJR UH  
VB0 VMK DA2R RL PPY JK  
PPHS1 VB0>NNL2 RG IW PPIS2  
NN1 VBDZ VBZ>VHD PPHO2>VBDR  
NPM2 CSW JB DB RGQ APP\$  
DDQ\$ PNQS NPM2 DDQV DDQ\$ MC2  
VM RG VDZ MF VMK>NN1  
MC-MC>VVGK NPM2 MC\$ JA II  
NP1>JJT RGR VBG DB2>NN  
JJT RGQ RGQV>NNO2 DB2>DA1  
PPX2>NNSB PPHO1>VM VDN>PPY  
VBZ>PNQO>JJ MF>BCS>NNU1  
MC>NNU>VVNK>PNQO>NNO1>XX  
RA>NNS>VMK>VHZ>VVZ>NPD2  
>NNL2>PPH1>NNL>NN1>DD>NNT  
IW>NNS2>APP\$>JBT>MC-MC>VDZ  
RR>NNS2>JK>VDG>MC2>DAR  
PPHS1>NNU2>JK>DB>RGQV>APP\$  
XX>RP>APP\$>NNJ1>NNU2>VD0  
CSA>DAR>PNQO>RR>NNU>VBN  
LE>PNQV\$>CS>CSN>MF>NNS2  
>NNT1>VVD>PNQV\$>NN1\$>APP\$>NNL2

VDN BCS MC1 MC2 RG NP2  
RGQV EX RGR VD0 PN1 \$  
VVZ RGQV MC RGA MC-MC DD1  
MC APP\$ JK NNU2 NNS2 VD0  
VHG NPD1 PPIS2 NNU NNO1 RT  
VHG IO NNSB NNO1 RT RRT  
CSW DD NN VHN RRR CCB  
VM ICS RGQ PNQVO \$>NNL  
VDG NN1\$ PP\$ NNO1 VVN NNO2  
MC1 PP\$ RR NNSB NP2 VH0  
ZZ2 PNQV\$ VVVK NN1\$ EX CF  
DA1 UH PNQVS VDZ DAR XX  
XX NNSA1 VDD RPK VDG VVVK  
VBM DAR ND1 NNJ1 NPM2 NN2  
VDD VVZ TO CCB PN1 PNQVO  
VHG NNSA2 TO VM NNO1 RT  
VHD EX NNJ1 DAR DA1 DAR  
TO CSN VBN PPHS2 NN NNSB2  
JBR VBM PNQS VDD ND1 PPIS1  
NN1\$ CSW JBR IF LE VBM  
PPIS2 RGT VDD NNJ VVN NPD2  
PNQS NNSA2 DD2 NNT1 VM NNSA1  
DD2 PPH1 MF VBN PNQS PPX1  
NNT APP\$ JK PNQV\$ IF NNU2  
JK DA2R NNSA1 RRT VM PPHS1  
JBT PNX1 DAR DD PPHS2 VB0  
MC NNU1 CF DD1 DDQ CC  
ND1 NNS AT PNX1 RRT VD0  
CS \$ PNX1 AT1 DD BCS  
VBR VBM II CC NNSB PNQS  
NPM2 IW NNO1 VDN PPIS1 NNU1  
PPHS1 VHD MC-MC NNJ2 PNX1 PPX2



VBG NNO1 MC PPIO2 PN1 RL  
 NP PPIS1 VDN NPD2 JBT CSA  
 RGA NNSB1 VH0 PPHO2 VDN JA  
 ICS NNU2>NNL MD PPY DDQ\$  
 RGA VHZ NNSB2 PPIS2 EX DD2  
 DD2 RR RR PNQO PPY PPIO2  
 NPD2 DAT VVNK PPIS2 NC2 PPH1  
 DA PPIS2 NP1 PNQO VDD NPM2  
 RPK VBM PPIO2 PN1>NNJ2 RL  
 DB2 VDG>NNL2 VVGK VHZ CSW  
 VBG JK VBZ CS JBR VMK  
 PPHS1 VDG DB MC1 VD0 MC-MC  
 NC2 APP\$ BTO JJT RRT NP  
 NNU1 JBT VVD NPM2 BTO PPY  
 ICS RRQ CF>NNJ2 NNU1 PPHS2  
 \$ CSN VBDZ PN1 AT DD2  
 IW>NNJ2 NNO1 RL>NNJ1 JB  
 CCB VBZ NPD1 VMK DD2 JB  
 NNSA1 VDD RA>JJR>JJT NNO1  
 RA RL DD2 CST VDN>VVN

*Test* consisted of the tag sequences:

AT>NN1>VVD II AT>NNL1 IO NP1 NP1 II NP1  
 >NNJ>NN1 II MC RA II NPD1 AT MD IO NP1  
 >NNJ>NN1 II MC RA II NPD1 AT MD IO NPM1  
 CCB RRQ VDZ PPHS1>VV0>NN1 IW PP\$ RR>JJ>NNS2  
 AT>JJ>JJ>NN1 II NP1>NNL1>VVD RP>NNT2 RA  
 CCB AT>JJ>NN1>VVZ VBZ PPH1 II>VVG AT>NN2  
 AT>JJ>NN1 AT NP1 VBZ AT>NN1 IO DA2>NN2  
 II NP1 AT>NN1 II>NN1 CC>NN1 VBZ RR>JJ  
 AT>NNS1>VVD II>PPIO1 II>JJ>NN2 PPY>VH0>NN1  
 RRR RP AT1>NNL1>RRQ EX VBDZ AT1>NN1>VVG RP  
 PPIS2>VVD II AT>NN1 CC>VVD ICS>JJ>IF NP1

AT NN2 AT NN1 VBDZ VVN RG RR VVD AT>NNL1  
PPHS2 VBDR RR VVN IW JJ NN2 CC JJ JJ NN2  
CCB CS DD NN2 VV0 AT1 JJ NN1 VVZ AT DA  
CF RR RL PPH1 VBZ CC II NP1 IO DB NN2  
PPHS1 VBZ VVN TO VV0 II>NNL1 IF MC NNT2  
AT>NN1 NN2 VH0 VVN AT1 JJ NNT1 II NPM1  
NNJ>NN2 VH0 VVN AT>NNJ IO VVG PPHO2>NN2  
EX VBZ RR>NN1 IF>NN2>NN2 II JJ NNT2  
AT1>NN1 VVD MD NNT1 II NP1 II>NNL1 NP1  
AT>NN1 VBDZ VVN RR MC>NNU2 II AT>NNL1  
PPHS1 VHD AT1 JJ>NN1>NN1>NN1 RL II NPD1  
PPIS1 VH0 RR RR VVN DDQ PPHS1 VDD IW PPHO2  
RR JJ CST NP1 VBZ AT1 JJ>NNL1 IF>NN2  
PPH1 VVZ PPH1 VM VB0 RG JJ TO VV0 PPHO2  
DB JJ>NN2 VH0 AT1 II AT>NN1>NN1>NN1  
PPHS1 RT VVD RP TO VV0 AT1 JJ>NNS1 RP  
MC CC MC NNT2 PPIS2 VVD JJ>NN1>NNU2 RL  
PPHS2 VBDR JJ CSA PPHS2 RR VBR IW AT>NN  
AT>NN1 II DD1 NPM1 NNT1 II NP1 VBDZ JJ  
DD1 VBZ AT1>NN1 IO RR II MC>NNO II AT>NNPM1>NN1  
CC AT JJ>NNJ IO>NN2 VV0 AT1>NN1 II AT>NN1>NNJ  
PPH1 VBZ VVN CST DB>NN2 VM VB0 RP II JJ>RRR RT  
EX VBZ AT>NN1 II AT>NNP1>NN1 II NP1 CC>NNP1>NNL1  
APP\$>NN1>NNP1>NNP1 VVD II AT>NN1>RR ICS AT>NN1>VVD  
IF DD1 PPIS2 VH0 RR TO VV0 II AT>JJ>NNP1>NN1>NN1  
AT>NN1 VBZ VVN AT>NNP1>CC>NNP1>NNL1 II APP\$>JJ>NN1  
PPIS1 VH0 RR VVN>RRQ>PPY>VV0>AT1>NN>IW>NN1>VVG>NN2  
AT>JJ>JJ>NN1 II>NN1>NN1 VBZ AT>NN1>IO AT>NNNT1  
NN1>CC>NN1>RT II MC>II>NNP1>VHZ>VVN>NN>AT>NNL1  
ICS>NN1>NN1>NN1 VBZ AT>MD>RGT>JJ>NN1>IF>JJ>NN2  
CCB>RL>NN2>VVD>AT1>JJ>NN1>PPY>VBR>RL>IF>AT>NNJ  
DDQ>VBZ>RRR>PPHS2>VVD>RR>JJ>JJ>CC>JJ>VVN>II>NN1

ICS DB PNQS VVZ NN1 NN1 CS PPY VM VV0 AT JJ NN1  
 RR MC1 IO AT NN2 CC NN2 PPIS1 VV0 IO VVG AT1 NN1  
 II APP\$ NN1 PPHS1 VVD PPX1 RR II AT1 NN1 II NN1  
 AT NN2 II AT NN1 VBDR RR VVN II NP1 JJ>NNL1  
 AT NP1 NN1 NN1 VVZ RP VVG JJ NN1 II>NNS1 NP1  
 PPH1 VBDZ APP\$ NN1 PPHS1 VVD TO VV0>NN2 II AT>NN1  
 AT>NN1 VBZ CST PPHS2 VV0 CC APP\$>NN2 RR VV0 RL  
 CF RG RR CSA>NN1 VBZ VVN NP1 VBZ AT1 JJ>NN1  
 PPHS1 RR VVZ TO VB0 VVG AT1>NN1 II AT>NN1>NN2  
 AT JJ>NN1 VBDZ CST AT>NN1 VBDZ JJ TO VV0>NN2  
 PPH1 VBZ VVN II DD>NN1 VVG>NN2 ICS AT RGT JJ  
 PPHS2 VV0 JJ>NN2 JJ>NN2 CC AT>NN1 IO AT>NN1  
 CSA PPIS2 VVD TO VV0>NNL1 AT1 JJ>NNJ>NN1 VVD RP  
 II>NNT1 AT NP1>NN1 VBDZ RP MC II MC RR  
 CCB>NN2 VVD RR ICS AT>NNJ1 IO NP1 \$>NN1  
 DD1 VBZ AT>JJT>NN1 IF DD>NNT1 II MC MC  
 AT JJ>NN1>NN1 II>NNT1 VBDZ RP MC II MC

*Yardstick* consisted of the following parse trees:

(S (N AT JJ JJ>NN1  
     (P II (N NP1>NNL1)))  
     (V (R VVD RP) (NP>NNT2 RA)))  
 (S (N AT JJ>NN1  
     (N AT NP1))  
     (V VBZ (N AT>NN1  
         (P IO (N DA2>NN2))))))  
 (S (P II (N NP1))  
     (S (N AT>NN1  
         (P II (N>NN1 CC>NN1)))  
         (V VBZ (J RR JJ)))  
     (S (N PPHS2  
         (V VBDR (J RR VVN)

(P IW (N (N& JJ NN2) CC  
 (N+ JJ JJ NN2))))))  
 (S (N AT NN1 (P II  
 (N DD1 NPM1 NNT1  
 (P II (N NP1))))))  
 (V VBDZ JJ))  
 (S (FA CCB  
 (FA CS (N DD NN2)  
 (V VVO)))  
 (N AT1 JJ NN1  
 (V VVZ (N AT DA))))  
 (S (J RR JJ (FN CST  
 (N NP1)  
 (V VBZ (N AT1 JJ>NNL1  
 (P IF>NN2))))))  
 (S (N PPHS2)  
 (V VBDR (J JJ))  
 (FA CSA (N PPHS2)  
 RR (V VBR (P IW (N AT>NN))))  
 (S CC (N AT JJ>NNJ  
 (P IO (N>NN2)))  
 (V VVO (N AT1>NN1  
 (P II (N AT>NN1>NNJ))))))  
 (S EX  
 (V VBZ (N AT>NN1)  
 (P II (N AT>NN1>NN1))  
 (P II (N (N&>NN1)  
 CC (N+>NN1>NNL1))))))  
 (S (N AT JJ>NN1  
 (P II (N>NN1>NN1)))  
 (V VBZ (N AT>NN1  
 (P IO (N AT>NN1>NN1))))))

```

(S (P II (N APP$ NN1))
  (S (N PPHS) (V VVD (N PPX1) RR
    (P II (N AT1 NN1 (P II (N NN1)))))))
(S (N PPH1
  (V VBDZ (N APP$ NN1) (SI (N PPHS1) (V VVD))
    (TI TO VVO (N NN2 (P II (N AT NN1))))))
(S (N PPHS) RR
  (V VVZ (TI TO VBO VVG
    (N AT1 NN1) (P II (N AT NN1 NN2))))))
(S (N AT JJ NN1)
  (V VBDZ (FN CST (N AT NN1)
    (V VBDZ (J JJ (TI TO VVO (N NN2)))))))

```

Finally, *Plausible* consisted of the following parse trees:

```

(S (N AT NN1)
  (V VVD (P II (N AT NNL1
    (P IO (N NP1 NP1
      (P II (N NP1)))))))
(N NNJ NN1 (P II MC RA)
  (P II (N NPD1 (N AT MD
    (P IO (N NP1))))))
(N (NNJ NN1 (P II MC RA)
  (P II (N NPD1 (N AT MD
    (P IO (N NPM1))))))
(S CCB
  (S (N AT JJ NN1)
    (V VVZ (V VBZ (N PPH)
      (P II (TG VVG
        (N AT NN2))))))
(S (N AT NNS1)
  (V VVD (P II (N PPI01))
    (P II (N JJ NN2))

```

(S (N PPY) (V VHO (N NN1))))  
 (S (N AT NN2 (S (N AT NN1)  
 (V VBDZ VVN RG RR)))  
 (V VVD (N AT>NNL1)))  
 (S (S& CF (S (S RR (S RL (N PPH1)  
 (V VBZ))))))  
 (S+ CC (P (P II (N NP1))  
 (P IO (N DB>NN2))))  
 (S (N PPHS1)  
 (V VBZ VVN (TI TO>VVO  
 (P II (N>NNL1))  
 (P IF (N MC>NNT2))))))  
 (S (N AT1>NN1) (V>VVD (NR MD>NNT1)  
 (P II (N NP1 (P II  
 (N>NNL1 NP1))))))  
 (S (N AT>NN1) (V>VBDZ VVN RR MC>NNU2  
 (P II (N AT>NNL1))))  
 (S (N PPHS1) (V>VHD (N AT1>JJ>NN1>NN1>NN1)  
 RL (P II (N>NPD1))))  
 (S (N>PPIS1)  
 (V>VHO RR RR>VVN (FN (N>DDQ)  
 (N>PPHS1)  
 (V>VDD  
 (P>IW (N>PPHO2))))))  
 (S (N>DB>JJ>NN2) (V>VHO  
 (N>AT1>II  
 (N>AT>NN1>NN1>NN1)))  
 (S>EX  
 (V>VBZ (N>AT>NN1) (P>II (N>AT>NN1>NN1))  
 (N>(N&>NP1) CC (N>+>NP1>NNL1))))  
 (S (N>AT>NN1)  
 (V>VBZ>VVN (N>(N&>AT>NN1)

CC (N+ NP1>NNL1  
(P II  
(N APP\$ JJ>NN1))))))

# Bibliography

- [1] Hiyan Alshawi, editor. *The CORE Language Engine*. The MIT Press, 1992.
- [2] Eric Atwell, John Hughes, and Clive Souter. AMALGAM: Automatic Mapping Among Lexico-Grammatical Annotation Models. In *Proceedings of ACL workshop on The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, page to appear, 1994.
- [3] J. K. Baker. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97<sup>th</sup> Meeting of the Acoustical Society of America*, pages 547–550. 1979.
- [4] Robert C. Berwick. *The acquisition of syntactic knowledge*. MIT Press, 1985.
- [5] Robert C. Berwick, G. Edwards Barton, Jr, and Eric Sven Ristad. *Computational Complexity and Natural Language*. MIT Press, 1987.
- [6] Ezra Black, Roger Garside, and Geoffrey Leech, editors. *Statistically driven computer grammars of English the IBM-Lancaster approach*. Rodopi, 1993.
- [7] Joan Bresnan, editor. *The Mental Representation of Grammatical Relations*. The MIT Press, 1978.
- [8] Eric Brill. *A Corpus-Based Approach to Language Learning*. PhD thesis, University of Pennsylvania, 1993.
- [9] Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. Deducing Linguistic Structure from the Statistics of Large Corpora. In *AAAI-92 Workshop Program: Statistically-Based NLP Techniques, San Jose, California*, 1992.



- [10] Ted Briscoe. Noun Phrases are Regular: a Reply to Professor Sampson. In W. Meijs, editor, *Corpus Linguistics and Beyond*. Rodopi, 1987.
- [11] Ted Briscoe and John Carroll. Generalised Probabilistic LR Parsing of Natural Language (Corpora) with Unification-based Grammars. Technical report number 224, University of Cambridge Computer Laboratory, 1991.
- [12] Ted Briscoe and Nick Waegner. Robust Stochastic Parsing Using the Inside-Outside Algorithm. In *Proceedings of the AAAI Workshop on Statistically-based Techniques in Natural Language Processing*, 1992.
- [13] Thomas M. Bruel. Language Modelling for a Real-World Handwriting Recognition Task. In *Computational Linguistics for Speech and Handwriting Recognition: A one-day workshop organized by L. J. Evett and T. G. Rose as part of the AISB 1994 Workshop Series*, 1994.
- [14] Wray Buntine. A Critique of the Valiant Model. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 837–842, 1989.
- [15] Glenn Carroll and Eugene Charniak. Two Experiments on Learning Probabilistic Dependency Grammars from Corpora. In *AAAI-92 Workshop Program: Statistically-Based NLP Techniques, San Jose, California*, 1992.
- [16] John Carroll. *Practical Unification-based Parsing of Natural Language*. PhD thesis, University of Cambridge, March 1993.
- [17] John Carroll and Ted Briscoe. Integrating probabilistic and knowledge-based approaches to corpus parsing. In *Computational Linguistics for Speech and Handwriting Recognition: A one-day workshop organized by L. J. Evett and T. G. Rose as part of the AISB 1994 Workshop Series*, pages 1–8, 1994.
- [18] John Carroll, Claire Grover, Ted Briscoe, and Bran Boguraev. A Development Environment for Large Natural Language Grammars. Technical report number 233, University of Cambridge Computer Laboratory, 1991.
- [19] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23, 1952.

- [20] Mahesh V. Chitrao and Ralph Grishman. Statistical Parsing of Messages. In *AAAI-92 Workshop Program: Statistically-Based NLP Techniques, San Jose, California*, pages 263–266, 1992.
- [21] Noam Chomsky. *Aspects of the Theory of Syntax*. The M.I.T Press, 1965.
- [22] Noam Chomsky. *Reflections on Language*. Pantheon, 1975.
- [23] Noam Chomsky. *Lectures on Government and Binding*. Dordrecht: Foris, 1981.
- [24] K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Language Processing, ACL*, 1988.
- [25] K. Church and R. L. Mercer. Introduction to the Special Issue on Computational Linguistics Using Large Corpora. *Computational Linguistics*, 19.1:1–23, 1993.
- [26] K. Church and R. Patil. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8:139–49, 1982.
- [27] Kenneth Ward Church. Book Review of *theory and practice in corpus linguistics*, edited by Jan Aarts and Willem Meijs. *Computational Linguistics*, 17.1:99–103, 1991.
- [28] D. Cutting, J. Kupiec, J. Pedersen, and P. Silbun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Language Processing, ACL*, 1992.
- [29] D.R. Dowty, R.E. Wall, and S. Peters. *Introduction to Montague Semantics*. D. Reidel Publishing Company, 1981.
- [30] T. Ellman. Explanation-based learning: a survey of programs and perspectives. *ACM Computing Surveys*, 21:163–222, 1989.
- [31] J Fain, J .G Carbonell, P. J Hayes, and S. N. Minton. MULTIPAR: A Robust Entity Oriented Parser. In *Proceedings of the 7<sup>th</sup> Annual Conference of The Cognitive Science Society*, 1985.

- [32] Tom E. Fawcett. Learning from Plausible Explanations. In *Proceedings of the 6<sup>th</sup> International Workshop on Machine Learning, Cornell University, Ithaca, New York*, pages 37–39, 1989.
- [33] Steven Finch. *Finding Structure in Language*. PhD thesis, Edinburg University, 1993.
- [34] J. Fodor, T. Bever, and M. Garrett. *The Psychology of Language: An Introduction to Psycholinguistics and Generative Grammar*. McGraw-Hill, 1974.
- [35] Janet Dean Fodor. Parameters and Parameter Setting in a Phrase Structure Grammar. In Lyn Frazier and Jill de Villiers, editors, *Language Processing and Language Acquisition*, pages 225–255. Kuwer Academic Publishers, 1990.
- [36] Sandiway Fong. The Computational Implementation of Principle-based Parsers. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 65–82. The MIT Press, 1991.
- [37] W. N. Francis and H. Kučera. Manual of information to accompany a Standard Sample of Present-day Edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University, 1979.
- [38] G. Gazdar, E. Klein, G.K. Pullum, and I.A. Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, 1985.
- [39] William A. Gale and Kenneth W. Church. Poor estimates of context are worse than none. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 283–287, 1990.
- [40] R. Garside, G. Leech, and G. Sampson, editors. *The Computational Analysis of English: A Corpus-based Approach*. Longman, 1987.
- [41] G. Gazdar and C. Mellish. *Natural Language Processing in Lisp*. Addison-Wesley, 1989.

- [42] E. M. Gold. Language Identification to the Limit. *Information and Control*, 10:447–474, 1967.
- [43] S. J. Graham and S. J. Rhodes. Practical syntactic error recovery in compilers. In *ACM Symposium on Principles of Programming Languages*, pages 52–58, 1973.
- [44] Richard H. Granger. The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-formed Text. *Computational Linguistics*, 9:188–196, 1983.
- [45] Claire Grover, Ted Briscoe, John Carroll, and Bran Boguraev. *The Alvey Natural Language Tools Grammar (Third Release)*. Technical report, University of Cambridge Computer Laboratory, 1992.
- [46] Robin Haigh, Geoffrey Sampson, and Eric Atwell. Project APRIL—a progress report. In *Proceedings of the 26<sup>th</sup> Annual Meeting of the Association for Computational Linguistics*, pages 104–112, 1988.
- [47] K Hammond and V.J. Rayward-Smith. A Survey on Syntactic Error Recovery and Repair. *Computer Language*, 9:51–67, 1984.
- [48] Zellig S. Harris. *Methods in Structural Linguistics*. University of Chicago Press, Chicago and London, 1951.
- [49] Philip Harrison, Steven Abney, Ezra Black, Dan Flickinger, Ralph Grishman, Claudia Gdaniec, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini, and Tomek Strzalkowski. Natural Language Processing Systems Evaluation Workshop, Technical Report rl-tr-91-362. In Jeannette G. Neal and Sharon M. Walter, editors, *Evaluating Syntax Performance of Parser/Grammars of English*, 1991.
- [50] D Haussler. Quantifying inductive bias: AI learning algorithms and Valiant’s learning framework. *Artificial Intelligence*, 36:177–221, 1988.
- [51] P. J. Hayes. Flexible Parsing. *Computational Linguistics*, 7.4:232–241, 1981.

- [52] P. J. Hayes and J. G Carbonell. Multi-strategy Construction-specific Parsing for Flexible Data Base Query and Update. In *Proceedings of the 7<sup>th</sup> International Conference on Artificial Intelligence*, pages 432–439, 1981.
- [53] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [54] John Hughes. *Automatically Acquiring a Classification of Words*. PhD thesis, University of Leeds, 1994.
- [55] Ray S. Jackendoff. *X-Bar Syntax: A Study of Phrase Structure*. The M.I.T Press, 1977.
- [56] Michael L. Maudlin Jaime G. Carbonell, W. Mark Boggs. XCALIBUR Project Report 1. Report cmu-cs-83-143, Carnegie-Mellon University, 1983.
- [57] Fred Jelinek. Self-organised language modelling for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 450–560. Morgan-Kaufmann, 1990.
- [58] K. Jensen, G. E. Heidorn, L. A. Miller, and Y. Ravin. Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness. *Computational Linguistics*, 9:147–160, 1983.
- [59] S. Johansson, G. Leech, and H. Goodluck. Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers. Technical report, Department of English, University of Oslo, 1978.
- [60] P. N. Johnson-Laird. *Mental Models*. Cambridge University Press, 1983.
- [61] Bernard E. M. Jones. Can Punctuation Help Parsing? In *15<sup>th</sup> International Conference on Computational Linguistics, Kyoto, Japan*, 1994.
- [62] G. J. F. Jones, H. LLOYD-Thomas, and J. H. Wright. Adaptive Statistical and Grammar Models of Language for Applications to Speech Recognition. In *Grammatical Inference Colloquium, Essex University*, 1993.

- [63] Robert T. Kasper and William C. Rounds. A logical semantics for feature structures. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 257–266, 1986.
- [64] Martin Kay. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Spärck Jones, and B. L. Webber, editors, *Readings in natural language processing*, pages 35–70. Morgan-Kaufmann, 1986.
- [65] Dennis Kibler and Pat Langley. Machine Learning as an Experimental Science. In *Proceedings of the 3<sup>rd</sup> European Working Session on Learning*,, 1988.
- [66] Jörg-Uwe Kietz and Sašo Džeroski. Inductive Logic Programming and Learnability. *SIGART Bulletin*, 5.1:22–32, 1994.
- [67] Kevin Knight. *Integrating Knowledge Acquisition and Language Acquisition*. PhD thesis, Carnegie Mellon University, August 1991.
- [68] Pat Langley and Jaime G. Carbonell. Language acquisition and machine learning. In Brian MacWhinney, editor, *Mechanisms of language acquisition*, pages 115–155. Lawrence Erlbaum Associates, 1987.
- [69] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56, 1990.
- [70] Fanny Leech. *An approach to probabilistic parsing*. MPhil Dissertation, 1987. University of Lancaster.
- [71] Geoffrey Leech and Roger Garside. Running a grammar factory: The production of syntactically analysed corpora or “treebanks”. In Stig Johansson and Anna-Brita Stenström, editors, *English Computer Corpora: Selected Papers and Research Guide*. Mouton de Gruyter, 1991.
- [72] James J. Lu, Monica D. Barback, and Lawrence J. Henschen. Interpreting Disjunctive Logic Programs Based on a Strong Sense of Disjunction. *Journal of Automated Reasoning*, 10:345–370, 1993.

- [73] John Lyons. *Introduction to Theoretical Linguistics*. Cambridge University Press, 1968.
- [74] D. Magerman and M. Marcus. Pearl: a probabilistic chart parser. In *Proceedings of the 2<sup>nd</sup> International Workshop on Parsing Technologies, Cancun, Mexico*, pages 193–199, 1991.
- [75] David Magerman and Carl Weir. Efficiency, Robustness and Accuracy in Picky Chart Parsing. In *Proceedings of the 30<sup>th</sup> ACL, University of Delaware, Newark, Delaware*, pages 40–47, 1992.
- [76] David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, February 1994.
- [77] M.P. Marcus. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, 1980.
- [78] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions of Programming Language Systems*, 4.2:258–282, 1982.
- [79] Chris S. Mellish. Some chart-based techniques for parsing ill-formed input. In *ACL Proceedings, 27<sup>th</sup> Annual Meeting*, pages 102–109, 1989.
- [80] M. Meteer, R. Schwartz, and R. Weischedel. Empirical studies in part of speech labelling. In *Proceedings of the fourth DARPA Workshop on Speech and Natural Language*, 1991.
- [81] R. Michalski. Pattern recognition as rule-guided inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:349–361, 1980.
- [82] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1.1:47–80, 1986.
- [83] T. M. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, 1978.
- [84] Balas K. Natarajan. *Machine Learning: a Theoretical Approach*. Morgan Kaufmann, 1991.

- [85] Sven Naumann and Jürgen Schrepp. Grammatical inference in DACS. In *Grammatical Inference Colloquium, Essex University*, 1993.
- [86] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga, 1980.
- [87] G. Nunberg. *The linguistics of punctuation*. Center for the Study of Language and Information, 1990.
- [88] Miles Osborne. *Text Parsing and the treatment of undergeneration*. First Year DPhil Report, 1992. University of York.
- [89] Miles Osborne and Derek Bridge. Inductive and deductive grammar learning: dealing with incomplete theories. In *Grammatical Inference Colloquium, Essex University*, 1993.
- [90] Miles Osborne and Derek Bridge. Learning unification-based grammars and the treatment of undergeneration. In *Workshop on Machine Learning Techniques and Text Analysis, Vienna, Austria*, 1993.
- [91] Miles Osborne and Derek Bridge. Learning unification-based grammars using the Spoken English Corpus. In *Grammatical Inference and Applications*, pages 260–270. Springer Verlag, 1994.
- [92] Miles Osborne and Derek Bridge. More for Less: Learning a Wide Coverage Grammar from a Small Training Set. In *International Conference on New Methods in Language Processing*. Centre for Computational Linguistics, UMIST, Manchester, 1994.
- [93] Dirk Ourston and Raymond Mooney. A Comprehensive Approach to Theory Refinement. In *Proceedings of the 8<sup>th</sup> National Conference on Artificial Intelligence*, pages 815–820, 1990.
- [94] Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30<sup>th</sup> ACL, University of Delaware, Newark, Delaware*, pages 128–135, 1992.
- [95] Steven Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.



- [96] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics: Volume 1: Fundamentals*. Center for the Study of Language and Information, 1987.
- [97] Geoffrey K. Pullum. On two recent attempts to show that english is not a CFL. *Computational Linguistics*, 10(3-4):182–186, 1984.
- [98] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1.1, 1986.
- [99] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1989.
- [100] Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A comprehensive grammar of the English language*. Longmans, 1985.
- [101] S Rajamoney and G DeJong. The Classification, Detection and Handling of Imperfect Theory Problems. In *Proceedings of the 10<sup>th</sup> International Joint Conference on Artificial Intelligence, Milan, Italy*, pages 205–207, 1987.
- [102] E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, 1991.
- [103] Dale W. Russell. *Language Acquisition in a Unification-based Grammar Processing System using a Real world Knowledge Base*. PhD thesis, University of Illinois at Urbana-Champaign, 1993.
- [104] G. Sampson. Evidence against the ‘Grammatical/Ungrammatical’ Distinction. In W. Meijs, editor, *Corpus Linguistics and Beyond*. Rodopi, 1987.
- [105] Geoffrey Sampson. Analysed corpora of English: a consumer guide. In Martha Pennington and Vace Stevens, editors, *Computers in Applied Linguistics*. Multilingual Matters, 1991.
- [106] Christer Samuelsson and Manny Rayner. Quantitative Evaluation of Explanation-Based Learning as an Optimisation Tool for a Large-Scale Natural Language System. In *Proceedings of the 12<sup>th</sup> International Joint Conference on Artificial Intelligence*, pages 609–615, 1991.
- [107] Antonio Sanfilippo. The (other) Cambridge ACQUILEX Papers. Technical report number 253, University of Cambridge Computer Laboratory, 1991.

- [108] R.C. Schank. Sam – A Story Understander. Research Report 43, Yale University Dept. of Computer Science, 1975. In collaboration with the Yale A.I Project.
- [109] C. Shannon. The mathematical theory of communication. *Bell System Technical Journal*, 27:398–403, 1948.
- [110] C. Shannon. Prediction and Entropy of Printed English. *Bell System Technical Journal*, 29:50–64, 1951.
- [111] E.Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.
- [112] R. A. Sharman, F. Jelinek, and R. L. Mercer. Generating a grammar for statistical training. In *Proceedings of the IBM Conference on Natural Language Processing*, 1988.
- [113] S. M. Shieber, H. Uszkoreit, F. C. N. Pereira, and M. Tyson. The Formalism and Implementation of PATR-II. In *Research on Interactive Aquisition and Use of Knowledge*. SRI International, Menlo Park, California, 1983.
- [114] Stuart M. Shieber. Evidence against the non-context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.
- [115] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. Center for the Study of Language and Information, 1986.
- [116] Steven Lawrence Small. *Word Expert Parsing: A theory of distributed word-based natural language understanding*. PhD thesis, University of Maryland, September 1980.
- [117] C. Souter and E.S Atwell. A Richly Annotated Corpus for Probabilistic Parsing. Report 92.13, University of Leeds School of Computer Studies, 1992.
- [118] C. Souter, T. O’Donoghue, and E.S Atwell. Training Parsers with Parsed Corpora. Report 91.2, University of Leeds School of Computer Studies, 1991.
- [119] Clive Souter. Systemic-Functional Grammars and Corpora. Report 89.12, University of Leeds School of Computer Studies, 1989.

- [120] Clive Souter and Tim O'Donoghue. *Probabilistic Parsing in the COMMUNAL Project*. Report 90.2, University of Leeds School of Computer Studies, 1990.
- [121] Irene Stahl, Birgit Tausend, and Rüdiger Wirth. Two Methods for Improving Inductive Logic Programming Systems. In *Machine Learning: ECML-93*, pages 41–55, 1993.
- [122] L. C. Taylor, C. Grover, and E. J. Briscoe. The Syntactic Regularity of English Noun Phrases. In *Proceedings, 4<sup>th</sup> European Association for Computational Linguistics*, pages 256–263, 1989.
- [123] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27:1134–1142, 1984.
- [124] Kurt Vanlehn and William Ball. A Version Space Approach to Learning Context-free Grammars. *Machine Learning*, 2.1:39–74, 1987.
- [125] W. M. W. Mark Boggs, J. G Carbonell, and I Monarch. The DYPAR-I Tutorial and Reference Manual. Technical report, Carnegie-Mellon University, 1983.
- [126] Ralph M. Weischedel. If the Parser Fails. In *18<sup>th</sup> Annual Meeting of the Association for Computational Linguistics, University of Pennsylvania*, page 95, 1980.
- [127] Ralph M. Weischedel. Meta-rules as a Basis for Processing Ill-formed Input. *Computational Linguistics*, 9:161–177, 1983.
- [128] Lydia White. *Universal Grammar and second language acquisition*. John Benjamins Publishing Company, 1989.
- [129] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.
- [130] W. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13.8, 1970.
- [131] Ave Wrigley. Parse Tree N-grams for Spoken Language Modelling. In *Grammatical Inference Colloquium, Essex University*, 1993.

- [132] S. J. Young and H. H. Shih. Computer Assisted Grammar Construction.  
In *Grammatical Inference and Applications*, pages 282–290. Springer Verlag,  
1994.
- [133] George K. Zipf. *The psycho-biology of language: an introduction to dynamic  
philology*. Routledge, 1936.